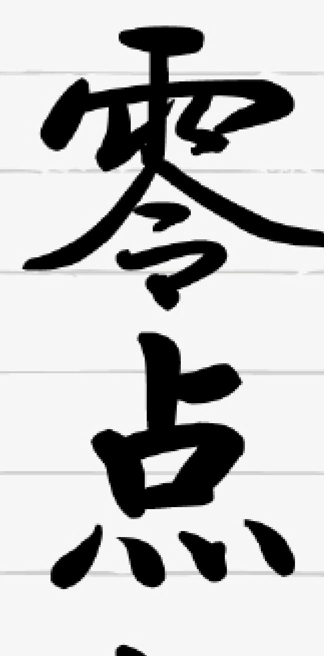



零点起飞学编程







飞



超值光盘，你值得拥有！

- 本书实例源文件
- 本书配套教学视频

CD-ROM

Excel VBA

9小时高清多媒体教学视频

魏汪洋　等编著

- **循序渐进：** 基础 → 进阶 → 实战
- **科学编排：** 基本语法 → 典型实例 → 编程练习 → 项目实战
- **学练结合：** **153**个实例、**2**个项目案例、**116**个练习题
- **视频讲解：** 提供配套**多媒体教学视频**
- **有问必答：** 提供**QQ群**、**E-mail**和**论坛**答疑服务

清华大学出版社

零点起飞学编程

零点起飞学 Excel VBA

魏汪洋　等编著

清华大学出版社
北　京

内 容 简 介

本书结合大量实例，由浅入深、循序渐进地介绍了 Excel VBA 开发技术。本书从初学者的角度出发，从最简单的 VBA 语法，一直讲解到 VBA 的数据库编程，图文并茂，通俗易懂，力图以最直观的方式使读者学好各个知识点。**本书特意提供了典型习题及教学 PPT，以方便教学。另外，本书还配了大量教学视频，帮助读者更好地学习。这些视频和书中的实例源代码一起收录于本书的配书光盘中。**

全书共 21 章，分 3 篇。第 1 篇为 Excel VBA 编程基础，主要介绍了 Excel VBA 的发展历史、Excel 2010 中的宏和开发 VBA 的环境、VBA 变量与语句、VBA 程序控制结构、VBA 数组和过程等。第 2 篇为 Excel VBA 进阶开发，主要介绍了 Excel VBA 的对象模型、Application 对象、工作簿对象、工作表对象和单元格对象、工作表界面、用户窗体、自定义 Excel 2010 功能区、图表对象、类模块和数据库编程等。第 3 篇为项目开发案例实战，主要介绍了教务管理系统和档案管理系统的实现方法。

本书适合没有编程基础的 Excel VBA 入门新手阅读，也可作为大中专院校及职业院校相关课程的教材。对于有一定基础的读者，可以通过本书进一步理解 VBA 的各个重要知识点。另外，本书也可供编程爱好者作为实际工作中的参考书籍。

图书在版编目（CIP）数据

零点起飞学 Excel VBA / 魏汪洋等编著. —北京：清华大学出版社，2013.7
（零点起飞学编程）
ISBN 978-7-302-31825-5

Ⅰ. ①零…　Ⅱ. ①魏…　Ⅲ. ①表处理软件　Ⅳ. ①TP391.13

中国版本图书馆 CIP 数据核字（2013）第 062966 号

责任编辑： 夏兆彦
封面设计： 欧振旭
责任校对： 胡伟民
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn, http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 25　　**字　　数：** 625 千字
版　　次： 2013 年 7 月第 1 版　　**印　　次：** 2013 年 7 月第 1 次印刷
印　　数： 1～ 000
定　　价： 　　元

产品编号：051520-01

前　　言

随着 Office 办公软件的流行，Excel 获得了广泛应用，Excel 在提供基本功能的同时，也提供了扩展 Excel 功能的工具，Excel VBA 是扩展 Office 办公软件功能的常用工具。

作为 Excel VBA 开发人员的必修课，笔者结合自己多年的开发信息系统经验和心得体会，精心编写了本书。本书以 Excel 2010 为载体，配合大量多媒体教学视频，通过系统的讲解和详细的示例，相信使初学者对 Excel VBA 开发有一个完整的认识。希望能在本书的引导下使读者掌握 Excel VBA 的开发方法。本书结合 Excel VBA 常用实例，深入浅出地讲解了 Excel VBA 开发环境、VBA 语句、Excel 对象模型和用户窗体的设计方法，并以大量实例贯穿于全书的讲解之中，最后还详细介绍了教务管理系统和档案管理系统的开发实例，使读者在实战中更深入地了解 Excel VBA 的开发。学习完本书后，读者可以具备独立开发 Excel VBA 的程序。

本书有何特色

1．配多媒体教学视频

本书提供配套的多媒体语音教学视频。在视频中对概念、示例和案例进行分析和详细地讲解，还加入了实际操作过程，旨在提高学习效率。

2．门槛低，容易入门

本书选取了 Excel VBA 开发的常见技术进行讲解，对读者的知识层次要求低，只要读者熟悉 Windows 操作系统即可。

3．内容全面、系统

本书详细介绍了 Excel VBA 开发所需要的知识，包括 VBA 语言基础、Excel 对象模型、用户窗体、数据库编程和自定义 Excel 界面等。

4．讲解由浅入深，循序渐进

本书的编排采用循序渐进的方式，内容梯度从易到难，讲解由浅入深，适合各个层次的读者阅读，并均有所获。

5．写作细致，处处为读者着想

本书概念讲解通俗易懂、代码注释详尽、程序运行步骤清晰，扫清了读者的学习障碍。

6．贯穿大量的开发实例和技巧

本书在讲解知识点时贯穿了大量短小精悍的典型实例，并给出了大量的开发技巧，力

求让读者获得真正实用的知识。

7．案例清晰

本书详细介绍了教务管理系统和档案管理系统的制作方法，详细介绍了 Excel 2010 中开发信息管理系统的一般方法。

8．提供教学PPT，方便老师教学

本书适合大中专院校和职业学校作为职业技能的教学用书，所以专门制作了教学 PPT，以方便各院校的老师教学时使用。

本书内容安排

第1篇　Excel VBA编程基础（第1～8章）

本篇主要内容包括：Excel VBA 的发展历史、功能、用途，Excel VBA 中的宏、开发 Excel VBA 的环境、VBA 变量和语句、VBA 程序控制结构、VBA 数组及 VBA 过程和函数。

通过本篇内容的学习，读者可以掌握 Excel VBA 的发展历史和开发环境及 Excel VBA 编程的基础知识。

第2篇　Excel VBA编程进阶（第9～19章）

本篇主要内容包括：Excel 对象模型、Application 对象、工作簿对象、工作表对象、单元格对象、Excel 图表对象、Excel 用户窗体设计、自定义 Excel 功能区、操作 Excel 图表对象、使用类模块和 VBA 数据库编程。

通过本篇的学习，读者可以掌握 Excel VBA 对象模型的使用方法、Excel VBA 用户窗体的设计方法和自定义 Excel 功能区的方法；另外还可以掌握 Excel 图表对象和 VBA 类模块的使用方法，以及使用 Excel VBA 进行数据库编程的方法。

第3篇　Excel VBA对象模型篇（第20～21章）

本篇主要内容包括：教务管理系统和档案管理的实现方法。通过本篇的学习，读者可以掌握使用 Excel VBA 开发信息系统的方法。

本书光盘内容

- ❑ 本书配套教学视频；
- ❑ 本书涉及的源代码。

本书读者对象

- ❑ Excel VBA 入门人员；
- ❑ Excel VBA 开发人员；

- Excel VBA 爱好者；
- Office 软件培训机构人员；
- 大中专院校的学生；
- 相关培训班的学员。

本书阅读建议

- 建议没有编程基础的读者从前至后顺次阅读，尽量不要跳跃。
- 书中的实例和示例建议读者都要亲自上机动手实践，学习效果更好。
- 课后习题都动手做一做，以检查自己对本章内容的掌握程度，如果不能顺利完成，建议回过头来重新学习一下本章内容。
- 学习每章内容时，建议读者先仔细阅读书中的讲解，然后再结合本章教学视频，学习效果更佳。

本书作者

全书由魏汪洋主编并编写了第 1 章，张明川编写了第 2~6 章，吴庆涛编写了第 7 章，郑瑞娟编写了第 9 章，徐翠霞编写了第 10~13 章，赵海霞编写了第 14~16 章，李冠峰编写了第 18 章，杨春蕾编写了第 8、17、19、20、21 章，王红艺编写了第 22 章。另外，陈世琼、陈欣、陈智敏、董加强、范礼、郭秋滟、郝红英、蒋春蕾、黎华、刘建准、刘霄、刘亚军、刘仲义、柳刚、罗永峰、马奎林、马味、欧阳昉、蒲军也参与了本书编写。马林参与了本书后期整理工作，全书由魏汪洋统编定稿。

阅读本书的过程中，若有任何疑问，可以发邮件到 bookservice2008@163.com，以获得帮助。

魏汪洋

目　　录

第 1 篇　Excel VBA 编程基础

第 1 章　VBA 概述（教学视频：7 分钟）……2
1.1　什么是 VBA……2
1.1.1　VBA 的产生及发展历史……2
1.1.2　VBA 的应用……2
1.2　第 1 个 VBA 程序……3
1.2.1　创建“Hello World”应用程序……3
1.2.2　执行“Hello World”程序及查看结果……4
1.2.3　使用 VBA 调试器……4
1.3　VBA 的功能及用途……5
1.3.1　VBA 的功能……5
1.3.2　VBA 的用途及常用开发工具……5
1.4　小结……6
1.5　本章习题……6
第 2 章　Excel 中的宏与 VBA（教学视频：18 分钟）……7
2.1　认识宏……7
2.1.1　什么是宏……7
2.1.2　理解宏的功能……8
2.2　操作 Excel 中的宏……8
2.2.1　录制宏……9
2.2.2　保存宏……10
2.2.3　执行宏……11
2.2.4　编辑宏……12
2.2.5　删除宏……13
2.3　加载宏……14
2.3.1　在 Excel 中加载宏……14
2.3.2　在 Excel 中卸载加载宏……15
2.3.3　在 Excel 中保存加载宏……16
2.3.4　Excel 中的其他加载宏……17

2.4 宏的安全性……18
2.4.1 通过信任中心设置宏的安全……18
2.4.2 通过信任中心启用被禁的宏……19
2.5 宏的数字签名……19
2.5.1 添加数字签名……20
2.5.2 使用数字签名……22
2.6 小结……24
2.7 本章习题……24
第 3 章 开发 VBA 的工具（教学视频：23 分钟）……25
3.1 Excel 中的 Visual Basic 编辑器……25
3.1.1 打开编辑器……25
3.1.2 剖析编辑器……26
3.2 常用编辑器窗口……27
3.2.1 使用工程窗口查看工程结构……27
3.2.2 使用属性窗口查看工程属性……29
3.2.3 使用代码窗口编辑调试 VBA 代码……30
3.2.4 使用立即窗口查看工程结果……31
3.2.5 使用对象浏览器窗口查看所有对象……32
3.3 使用编辑器的代码输入功能……33
3.3.1 显示常用的属性和方法……34
3.3.2 显示参数……35
3.3.3 使用快速信息……35
3.4 调试 VBA……36
3.5 小结……38
3.6 本章习题……38
第 4 章 VBA 变量和运算符（教学视频：19 分钟）……40
4.1 认识常量……40
4.1.1 定义系统常量……40
4.1.2 自定义常量……41
4.2 使用变量……42
4.2.1 在 VBA 中声明变量……42
4.2.2 VBA 强制声明变量……44
4.2.3 VBA 变量的作用域……44
4.2.4 详解 VBA 变量的生存周期……46
4.3 使用运算符和表达式……47
4.3.1 算术运算符与算术表达式……47
4.3.2 比较运算符与比较表达式……48
4.3.3 逻辑运算符与逻辑表达式……49
4.4 小结……50
4.5 本章习题……50

第 5 章 VBA 语句（ 教学视频：15 分钟）……51
5.1 VBA 中的语句……51
5.1.1 什么是语句……51
5.1.2 使用赋值语句……53
5.1.3 使用注释语句……54
5.2 数据的输入和输出……56
5.2.1 输入对话框……56
5.2.2 提示对话框……57
5.2.3 显示程序运行结果……59
5.3 程序的中断……60
5.3.1 暂停程序……60
5.3.2 停止程序……60
5.4 小结……61
5.5 本章习题……61
第 6 章 VBA 程序控制结构（ 教学视频：47 分钟）……63
6.1 使用 VBA 选择结构……63
6.1.1 程序的结构……63
6.1.2 使用条件表达式……64
6.1.3 If…Then 语句……65
6.1.4 If…Then…Else 语句……66
6.1.5 IIf 函数……67
6.1.6 If…Then…ElseIf 语句……68
6.1.7 Select Case 语句……70
6.1.8 被嵌套的选择结构……73
6.2 使用 VBA 循环结构……75
6.2.1 For…Next 语句……75
6.2.2 For Each…In Next 语句……77
6.2.3 Do…Loop 语句……78
6.2.4 Until 型 Do…Loop 语句……80
6.2.5 While…Wend 语句……81
6.2.6 被嵌套的循环结构……83
6.3 使用其他控制语句……84
6.3.1 With 语句……84
6.3.2 Exit 语句……85
6.3.3 GoTo 语句……86
6.4 异常处理语句……88
6.4.1 On Error 语句……88
6.4.2 Resume 语句……89
6.5 小结……91
6.6 本章习题……91

第 7 章　使用 VBA 数组（教学视频：13 分钟）……93
7.1　什么是数组……93
7.1.1　数组的概念……93
7.1.2　声明一维数组……94
7.1.3　声明二维数组……95
7.2　静态数组……96
7.2.1　初始化静态数组……96
7.2.2　使用二维静态数组……97
7.3　动态数组……98
7.3.1　声明动态数组……98
7.3.2　定义数组大小……98
7.3.3　复制数组……99
7.3.4　清空数组或重定义数组……100
7.4　小结……101
7.5　本章习题……102
第 8 章　使用过程与函数（教学视频：38 分钟）……103
8.1　什么是过程……103
8.1.1　初识 VBA 模块……103
8.1.2　理解过程……104
8.2　VBA 中的 Sub 过程……104
8.2.1　创建 Sub 过程……105
8.2.2　调用 Sub 过程……106
8.3　VBA 中参数的传递……108
8.3.1　使用地址参数传递……108
8.3.2　使用值参数传递……109
8.3.3　使用数组参数传递……110
8.3.4　使用可选参数……111
8.3.5　使用可变参数……114
8.4　使用 Function 过程……116
8.4.1　理解函数……116
8.4.2　定义 Function 过程……119
8.4.3　直接调用 Function 过程……119
8.4.4　在 Excel 工作表中调用 Function 函数……120
8.5　小结……122
8.6　本章习题……123

第 2 篇　Excel VBA 编程进阶

第 9 章　对象模型（教学视频：22 分钟）……126
9.1　认识 Excel 对象……126

9.1.1 理解对象的属性 126
9.1.2 理解对象的方法 127
9.1.3 理解对象的事件 128
9.2 使用对象变量和对象数组 130
9.2.1 使用对象变量 130
9.2.2 使用对象数组 132
9.3 使用 Excel 集合对象 132
9.4 学习 Excel 2010 中的对象模型 134
9.4.1 理解 Excel 对象模型 134
9.4.2 了解 Excel 对象层次结构 135
9.5 小结 136
9.6 本章习题 136
第 10 章 Application 对象（教学视频：41 分钟） 138
10.1 常用操作 138
10.1.1 在 Excel 中使用“打开”对话框 138
10.1.2 实现 Excel 定时操作 139
10.1.3 退出 Excel 应用程序 140
10.1.4 在 Excel 过程中调用宏 141
10.1.5 激活 Office 2010 应用程序 142
10.1.6 获取 Excel 系统信息 142
10.1.7 为 Excel 操作指定快捷键 143
10.2 设置 Excel 界面外观 144
10.2.1 设置 Excel 界面标题栏 144
10.2.2 使用 Excel 界面状态栏 145
10.2.3 设置 Excel 窗口最大化和最小化 146
10.2.4 设置 Excel 界面光标形状 147
10.3 操作 Excel 单元格 148
10.3.1 快速选择 Excel 单元格 148
10.3.2 同时选择 Excel 多个区域单元格 149
10.3.3 取消复制或剪切操作 150
10.4 操作 Excel 文件 151
10.4.1 获取文件夹中指定文件的 Excel 文件名 151
10.4.2 获取 Excel 文件保存位置 153
10.4.3 打开最近使用的 Excel 文件 154
10.5 小结 155
10.6 本章习题 155
第 11 章 工作簿对象（教学视频：29 分钟） 157
11.1 引用 Excel 工作簿对象 157
11.1.1 引用 Excel 工作簿的方法 157
11.1.2 激活 Excel 工作簿对象 158

11.2 新建和打开 Excel 工作簿 ········ 159
11.2.1 新建 Excel 工作簿 ········ 159
11.2.2 打开 Excel 工作簿 ········ 160
11.3 保存 Excel 工作簿 ········ 161
11.3.1 使用 Save 方法保存 Excel 工作簿 ········ 161
11.3.2 使用 SaveAs 方法保存 Excel 文档 ········ 162
11.4 保护 Excel 工作簿 ········ 163
11.4.1 设置 Excel 工作簿打开密码 ········ 164
11.4.2 设置 Excel 工作簿保护密码 ········ 164
11.5 使用 Excel 工作簿事件 ········ 165
11.5.1 启用或禁用事件 ········ 166
11.5.2 使用 Excel 工作簿的窗口大小更改事件 ········ 167
11.5.3 使用 Excel 工作簿的打开事件 ········ 167
11.5.4 使用 Excel 工作簿的工作表激活事件 ········ 168
11.5.5 使用 Excel 工作簿的关闭之前事件 ········ 169
11.6 小结 ········ 170
11.7 本章习题 ········ 170

第 12 章 工作表对象（ 教学视频：48 分钟） ········ 172

12.1 引用 Excel 工作表对象 ········ 172
12.1.1 使用名称引用 Excel 工作表 ········ 172
12.1.2 使用索引号引用 Excel 工作表 ········ 173
12.2 新建和删除 Excel 工作表 ········ 174
12.2.1 新建 Excel 工作表 ········ 174
12.2.2 删除 Excel 工作表 ········ 175
12.3 选取和隐藏 Excel 工作表 ········ 176
12.3.1 选择 Excel 工作表 ········ 176
12.3.2 隐藏 Excel 工作表 ········ 177
12.4 复制和移动 Excel 工作表 ········ 179
12.4.1 复制 Excel 工作表 ········ 179
12.4.2 移动 Excel 工作表 ········ 180
12.5 打印 Excel 工作表 ········ 181
12.6 工作表的其他操作 ········ 183
12.6.1 设置 Excel 工作表的滚动区域 ········ 183
12.6.2 查看 Excel 工作表中的批注 ········ 184
12.6.3 删除空白 Excel 工作表 ········ 185
12.7 使用 Excel 工作表事件 ········ 186
12.7.1 使用 Excel 工作表激活事件 ········ 186
12.7.2 使用 Excel 单元格更改事件 ········ 187
12.7.3 使用 Excel 工作表的选择区域变化事件 ········ 188
12.7.4 使用 Excel 工作表右击事件 ········ 189
12.8 小结 ········ 190

12.9 本章习题 191

第 13 章 单元格对象（教学视频：36 分钟） 193

13.1 引用 Excel 单元格 193

13.1.1 引用 Excel 单元格 193

13.1.2 引用 Excel 单元格区域 195

13.1.3 使用偏移方式引用 Excel 单元格 197

13.2 操作 Excel 单元格 198

13.2.1 删除 Excel 单元格 198

13.2.2 清除 Excel 单元格内容 199

13.2.3 插入和隐藏 Excel 单元格 200

13.2.4 复制 Excel 单元格数据 201

13.2.5 保护 Excel 单元格 202

13.3 查找单元格数据 203

13.3.1 查找单个条件的数据 204

13.3.2 查找多个条件的数据 205

13.3.3 使用 Match 方法查找数据 206

13.3.4 筛选符合条件的数据 208

13.3.5 按颜色筛选数据 210

13.4 设置 Excel 单元格格式 212

13.4.1 设置 Excel 单元格边框 212

13.4.2 使用 Excel 条件格式 213

13.4.3 使用 Excel 数据条 215

13.4.4 自动排列前 10 名数据 215

13.5 小结 216

13.6 本章习题 217

第 14 章 工作表界面（教学视频：36 分钟） 219

14.1 认识 Excel 表单控件 219

14.2 使用 Excel 表单控件 220

14.2.1 “数值调节按钮”控件 220

14.2.2 “单选按钮”控件和“分组框”控件 222

14.2.3 “组合框”控件 224

14.2.4 “列表框”控件 226

14.2.5 “复选框”控件 227

14.2.6 “标签”控件 229

14.2.7 “按钮”控件 229

14.3 使用 Excel ActiveX 控件 231

14.3.1 添加控件和修改属性 231

14.3.2 为控件添加程序代码 234

14.4 小结 238

14.5 本章习题 238

第 15 章　自定义 Excel 用户窗体（教学视频：61 分钟）…………240
15.1　使用 Excel 窗体…………240
15.1.1　添加用户窗体…………240
15.1.2　设置窗体的属性和事件…………241
15.1.3　使用窗体的方法…………243
15.2　使用 Excel 控件…………245
15.2.1　认识控件…………245
15.2.2　设置窗体控件…………246
15.3　使用 Excel 标准控件…………249
15.3.1　“标签”控件…………249
15.3.2　“文本框”控件…………250
15.3.3　“复选框”控件和“单选按钮”按钮控件…………253
15.3.4　“列表框”控件…………257
15.3.5　“组合框”控件…………260
15.3.6　“图像”控件和“数字调节钮”控件…………261
15.3.7　TabStrip 控件和“多页”控件…………263
15.3.8　Refedit 控件…………266
15.4　使用 Excel 附加的 ActiveX 控件…………268
15.4.1　ListView 控件…………268
15.4.2　ImageList 控件…………272
15.4.3　TreeView 控件…………275
15.5　小结…………281
15.6　本章习题…………281
第 16 章　自定义 Excel 2010 功能区（教学视频：22 分钟）…………283
16.1　什么是 Open XML…………283
16.2　了解 Excel 2010 的功能区…………284
16.2.1　使用 Excel 2010 功能区的基本控件…………285
16.2.2　使用 Excel 2010 功能区的容器控件…………288
16.3　使用 Open XML 格式文件自定义功能区…………291
16.4　使用 UI 编辑器设计功能区…………294
16.5　小结…………297
16.6　本章习题…………297
第 17 章　控制图表（教学视频：16 分钟）…………299
17.1　引用 Excel 图表对象…………299
17.2　创建 Excel 图表对象…………300
17.2.1　使用 Excel 图表对象的常见属性…………300
17.2.2　添加 Excel 图表对象…………301
17.2.3　使用 Excel 图表对象的事件…………303
17.3　操作 Excel 图表…………305
17.3.1　操作图表区…………306

17.3.2 操作绘图区······307
17.3.3 操作坐标轴······308
17.3.4 操作数据系列······310
17.4 小结······311
17.5 本章习题······312

第 18 章 类模块（教学视频：13 分钟）······314
18.1 使用对象类······314
18.1.1 创建对象类······314
18.1.2 声明类模块中的对象······315
18.2 使用对象属性······316
18.2.1 使用变量创建属性······316
18.2.2 使用属性过程······317
18.3 创建类的方法······318
18.4 类的事件······319
18.4.1 创建事件的语法······320
18.4.2 创建事件的案例······320
18.5 小结······322
18.6 本章习题······322

第 19 章 数据库编程（教学视频：23 分钟）······324
19.1 认识数据库······324
19.2 使用 ADO 操作数据······325
19.2.1 创建 Connection 连接对象······325
19.2.2 使用 Recordset 记录集对象······326
19.2.3 获取数据库中的数据······329
19.2.4 在数据库中添加和删除记录······332
19.3 查询数据库中的数据······336
19.4 小结······340
19.5 本章习题······341

第 3 篇 项目开发案例实战

第 20 章 教务管理系统（教学视频：27 分钟）······344
20.1 设计功能······344
20.1.1 功能简介······344
20.1.2 设计思路······346
20.2 设计用户界面······346
20.2.1 创建 Excel 2010 工作表······346
20.2.2 设计教务管理登录界面······347
20.3 学生查询分数······348

20.4 教师查询分数 352
20.4.1 创建查询表 353
20.4.2 实现分数查询 357
20.5 设置操作权限 361
20.5.1 设定教师权限 361
20.5.2 设置管理员权限 362
20.6 退出程序 363
20.7 小结 364
第 21 章 档案管理系统（教学视频：27 分钟） 366
21.1 设计功能 366
21.1.1 功能简介 366
21.1.2 设计思路 367
21.2 设计用户界面 367
21.2.1 添加信息录入控件 367
21.2.2 添加控制按钮控件 369
21.3 实现程序功能 372
21.3.1 实现界面初始化功能 372
21.3.2 实现添加人事信息功能 373
21.3.3 实现查询和修改数据功能 376
21.3.4 实现退出程序和查看工作表 379
21.4 小结 382

第 1 篇　Excel VBA 编程基础

⏭ 第 1 章　VBA 概述

⏭ 第 2 章　Excel 中的宏与 VBA

⏭ 第 3 章　开发 VBA 的工具

⏭ 第 4 章　VBA 变量和运算符

⏭ 第 5 章　VBA 语句

⏭ 第 6 章　VBA 程序控制结构

⏭ 第 7 章　使用 VBA 数组

⏭ 第 8 章　使用过程与函数

第 1 章　VBA 概述

Excel 是 Microsoft 公司的办公软件 Office 中的一款，它以优秀的数据录入功能和强大的数据处理及分析功能而得到了广大办公人员的青睐，大大提升了他们的工作效率。Excel 可以满足大部分的日常数据处理需要，如果能够掌握 VBA 语言，就能够实现对数据的更高级的处理和操作。在 Excel 中使用 VBA，能够高效率地实现数据处理的自动化，将工作人员从简单而重复的数据处理工作中解放出来。本章将介绍的主要内容和学习目标有：

- 了解 VBA 的产生及发展历史；
- 了解 VBA 的概念；
- 掌握创建 VBA 程序的方法；
- 掌握 VBA 程序的调试器；
- 掌握 VBA 的功能及用途。

1.1　什么是 VBA

VBA 的英文全称是 Visual Basic For Application，即新一代标准宏语言，它是一种编程通用的自动化语言。

1.1.1　VBA 的产生及发展历史

在 VBA 产生之前，一些应用软件如 Excel、Word 等都有各自的编程语言供用户进行再开发使用，但每种语言都各不相同、并且互不兼容，需要用户针对不同的应用软件学习各自的编程语言，这样就使得应用软件在程序上不能互联。VBA 的产生就解决了这个问题。

VBA 是基于 Visual Basic for Windows 发展而来的，是 Visual Basic 的子集，而 Visual Basic 是由 Basic 发展而来的第 4 代编程语言。VBA 不但继承了 VB 的开发机制，而且 VBA 与 VB 有着相似的语言结构和开发环境。

1.1.2　VBA 的应用

VBA 是 Microsoft 公司长期追求的目标，使可编程应用软件得到完美的实现，它作为一种通用的宏语言能被所有的 Microsoft 可编程应用软件所共享，因此 VBA 主要用于 Microsoft 公司的办公软件，例如 Excel、Word、Access、PowerPoint、Outlook、Project 等一系列办公软件中。

1.2　第 1 个 VBA 程序

下面是一个简单的 VBA 控制台程序，从这里开始 VBA 的学习之旅，这一节主要学习创建 VBA 程序、运行 VBA 程序和调试 VBA 程序。

1.2.1　创建“Hello World”应用程序

【范例 1-1】 此程序是一个简单的 VBA 应用程序，程序运行后，输出“Hello World”。

（1）选择“开始|所有程序→Microsoft Office→Microsoft Office Excel 2010”命令，打开 Microsoft Excel 2010 窗口。

（2）选择“开发工具”选项卡。

说明： 若 Excel 2010 窗口的选项卡上没有开发工具选项卡，可选择“文件→选项”命令，弹出“Excel 选项”对话框，单击左侧列表上的“自定义功能区”列表项，在“自定义功能区”下拉列表中选择“主选项卡”列表项，然后在“主选项卡”的列表中选中“开发工具”复选框，使其打上“√”，如图 1.1 所示，设置完成后，单击“确定”按钮即可。

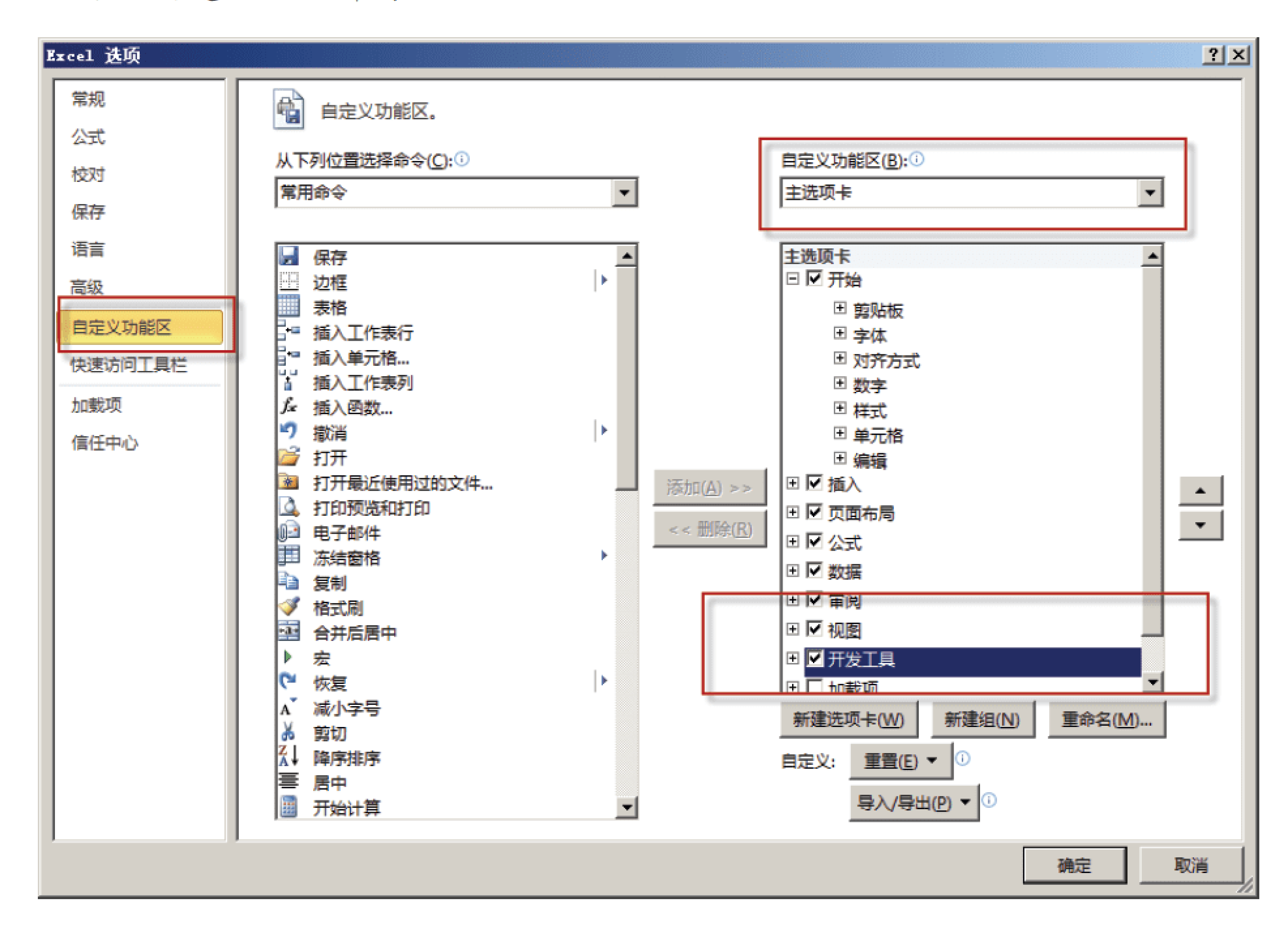


图 1.1　Excel 选项设置

（3）在“代码”组，单击“Visual Basic”按钮，弹出“Microsoft Visual Basic”编辑器的窗口。

（4）在该窗口中的“工程”子窗口中的 Sheet1 子节点上右击。

（5）依次选择“插入→模块”命令，弹出模块编辑窗口，如图 1.2 所示。

（6）在模块窗口中编辑如下代码，如图 1.3 所示。

```
01  Sub 第 1 个 VBA 程序()
02     MsgBox "Hello World!", vbOKOnly, "VBA 控制台程序"      '输出程序信息
03  End Sub
```

【代码解析】第 1 行和第 3 行是程序的开始和结尾，用于一个模块的始末，第 2 行是输出语句，用于实现输出 Hello World。

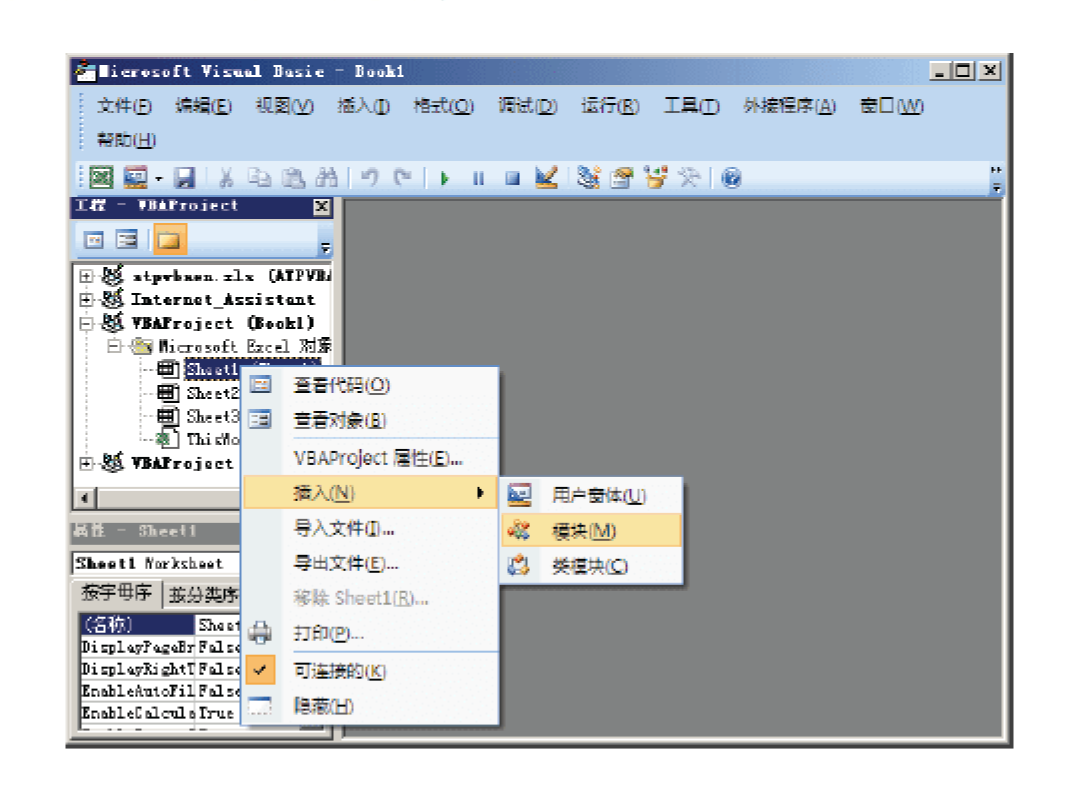


图 1.2　弹出模块编辑窗口

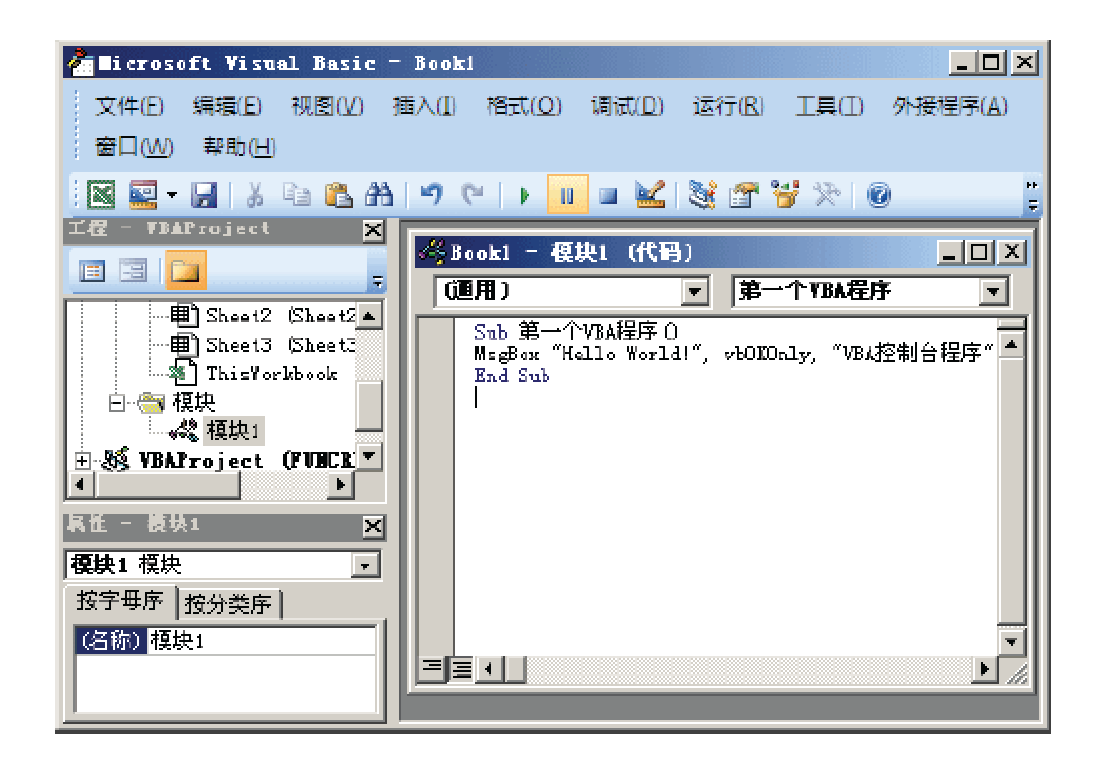


图 1.3　代码编辑窗口

1.2.2　执行“Hello World”程序及查看结果

（1）代码编辑完成后，在“Microsoft Visual Basic”窗口上，选择“运行→运行子过程/用户窗体”命令，开始运行程序。

（2）运行 VBA 程序，执行结果如图 1.4 所示。



图 1.4　Hello World 执行效果图

1.2.3　使用 VBA 调试器

在 Visual Basic 编辑器的窗口中选择“调试”命令，弹出的菜单中所列出的各个菜单项即为调试过程中所使用的工具，可以逐语句执行调试，也可以设置断点进行调试，同时可查看各变量的值。

例如：逐语句执行程序的调试方法。

（1）选择“调试→逐语句”命令，程序进入单步执行状态，在代码编辑窗上，第 1 行程序代码的前端出现了一个黄色的箭头；如图 1.5 所示。

（2）按下“F8”键，逐语句执行程序。

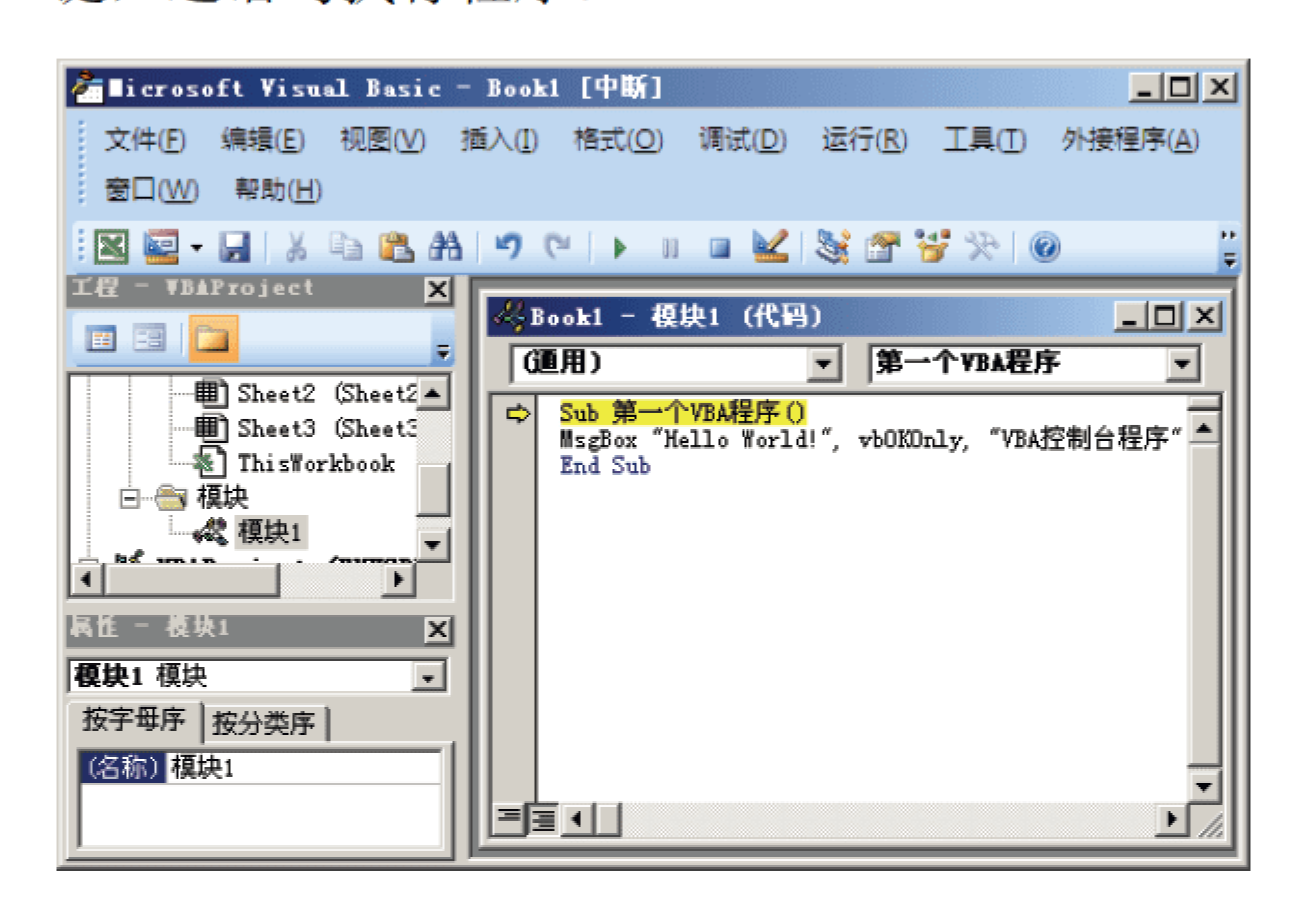


图 1.5　逐语句执行效果图

1.3　VBA 的功能及用途

VBA 作为新一代标准的宏语言，其学习容易、功能强大、通用性强，在实际工作中具有广泛的用途。

1.3.1　VBA 的功能

VBA 的功能主要体现在以下几个方面：

- 通过使用 VBA，可将重复性的任务自动化；
- 使用 VBA 可以自定义 Office 办公软件的工具栏、菜单和界面；
- 使用 VBA 可简化模板的使用；
- 使用 VBA 可借助已有办公软件，使其作为新的开发平台；
- 使用 VBA 可以依据实际工作的需要创建报表；
- 使用 VBA 可以对数据进行复杂的操作和分析。

1.3.2　VBA 的用途及常用开发工具

VBA 主要应用于 Excel、Word、Access、PowerPoint、FoxPro 等各种办公软件中，创建各种不同的解决方案。因此，对于在工作中需要经常使用 Office 套装软件的用户，学用 VBA 有助于使工作自动化，提高工作效率。另外，由于 VBA 可以直接应用 Office 套装软件的各项强大功能，所以对于程序设计人员的程序设计和开发更加方便快捷。

VBA 常用的开发工具主要是 Microsoft Visual Basic 编辑器，Microsoft Visual Basic 编辑器是一种可视化编辑器，其中包含了标题栏、菜单栏、工具栏、工程资源管理器、属性窗口、代码窗口等各种图形化工作界面，使工作界面更加美观、人性化。

1.4　小　　结

本章介绍了 VBA 的基本概念，即 Visual Basic For Application，即新一代标准宏语言，它是一种编程通用的自动化语言。介绍了 VBA 产生的历史背景和发展过程，并通过一个简单的 Excel VBA 程序展示了 Excel VBA 程序的创建、运行和调试方法，简要介绍了 VBA 的主要功能和用途。作为新一代标准宏语言，VBA 语言在 Excel、Word、Access、PowerPoint 可以使用，因此，在实际使用过程中具有强大的通用性。

1.5　本 章 习 题

1．VBA 的全称是________。

2．VBA 常用的开发工具是________。

3．下列哪些不是 VBA 的功能？（　　）

A．通过使用 VBA，可将重复性的任务自动化。

B．使用 VBA 可以自定义 Office 办公软件的工具栏、菜单和界面。

C．使用 VBA 使用模板的应用复杂化。

D．使用 VBA 可借助已有办公软件，使其作为新的开发平台。

4．VBA 语言可在下列哪些办公软件中使用？（　　）

A．Microsoft Word　　B．Microsoft Excel　　C．WPS　　D．Open office

5．VBA 的主要功能是什么？

6．VBA 主要应用在哪些方面？

7．创建自己的第一个 Excel VBA 程序，使程序输出：“我的第一个 VBA 程序！”。

第 2 章　Excel 中的宏与 VBA

宏是 Excel 2010 中的一个重要功能，VBA 是宏的实现途径，通过宏能够记录重复性操作，使工作人员能够从重复的劳动中解脱出来；VBA 是记录宏操作的语言，能够实现更加灵活的功能，能够对 Excel 中现有的功能实现扩展，还可以实现一些 Excel 中所不具备的功能。本章将介绍的主要内容和学习目标有：

- 了解宏的概念，掌握宏的两种创建方法；
- 掌握宏的基本操作；
- 掌握加载宏的使用方法；
- 了解宏的安全设置，能够通过宏安全选项的设置来保护文档；
- 了解数字签名的使用方法，能够使用数字签名来保护文档和宏。

2.1　认　识　宏

Excel 的一个最大的优势就在于能够创建和使用宏。通过宏的使用，能够实现对 Excel 的控制，扩展 Excel 的功能，提高效率。更为重要的是，宏是学习 VBA 最为有效的入门工具。本节将介绍 Excel 中宏的概念以及使用的有关知识。

2.1.1　什么是宏

对于初学者来说，宏也许是一个让人倍感高深的概念。按照微软的解释，宏是一系列存储于 Visual Basic 模块中的命令和函数，在需要执行时可以随时运行。通俗地说，宏是一系列能够自动完成某个任务的一组指令的集合。

在工作中，会经常需要完成某些重复的工作，例如，在制作人事档案表时，需要在表格中输入固定格式的地址和人员的名单。每次输入这些内容，都需要根据内容来设置格式，这将是一件很繁琐的重复工作。此时，可以将录入操作和需要设定的格式记录下来，在以后的操作中，只需要启动宏，操作就可以自动依次执行，从而节省用户的大量时间，极大地提高了工作效率。

使用宏，有其固有的优势。宏小巧，能够被分配给按钮、图形和控件等对象。因此，宏的执行方便而具有交互能力。对于不需要的宏，可以直接将其删除。同时，可以针对不同的操作要求来对宏进行组合，以实现操作的重组，从而快速准确地完成工作。正是因为宏的上述特点，无论对于初学者还是 Excel 高级用户，宏都是扩展 Excel 2010 功能和提高操作效率的有效工具。

提示： 宏之所以能够自动执行一系列的操作，是因为宏实际上是一组 VBA 程序代码。在录制宏的过程中，Excel 将操作转换为 VBA 代码保存在宏中。在运行宏时，Excel 驱动这些代码自动完成记录的操作。

以下面这个名为“填数字”的宏为例，这段宏在打开工作表的第一行中输入汉字数字，效果如图 2.1 所示。这段宏对应的是一段 VBA 程序，在 Visual Basic 编辑器的“代码”窗口中可以查看其代码，如图 2.2 所示。

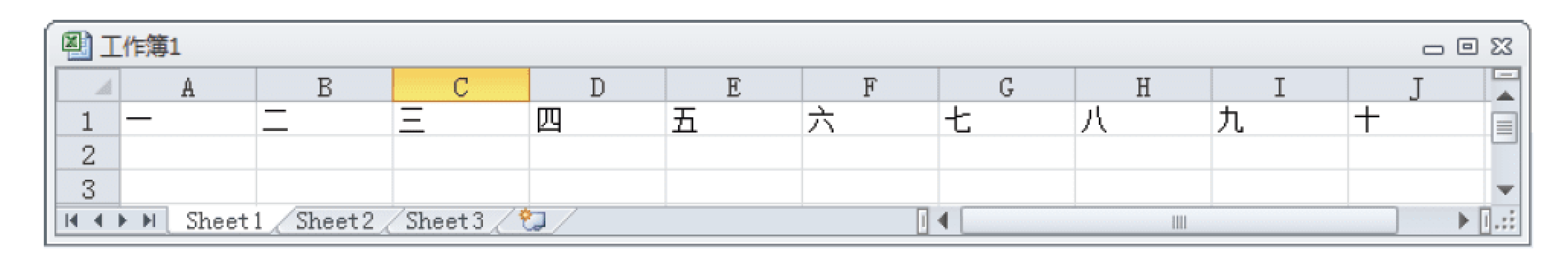


图 2.1　“填数字”宏的运行结果

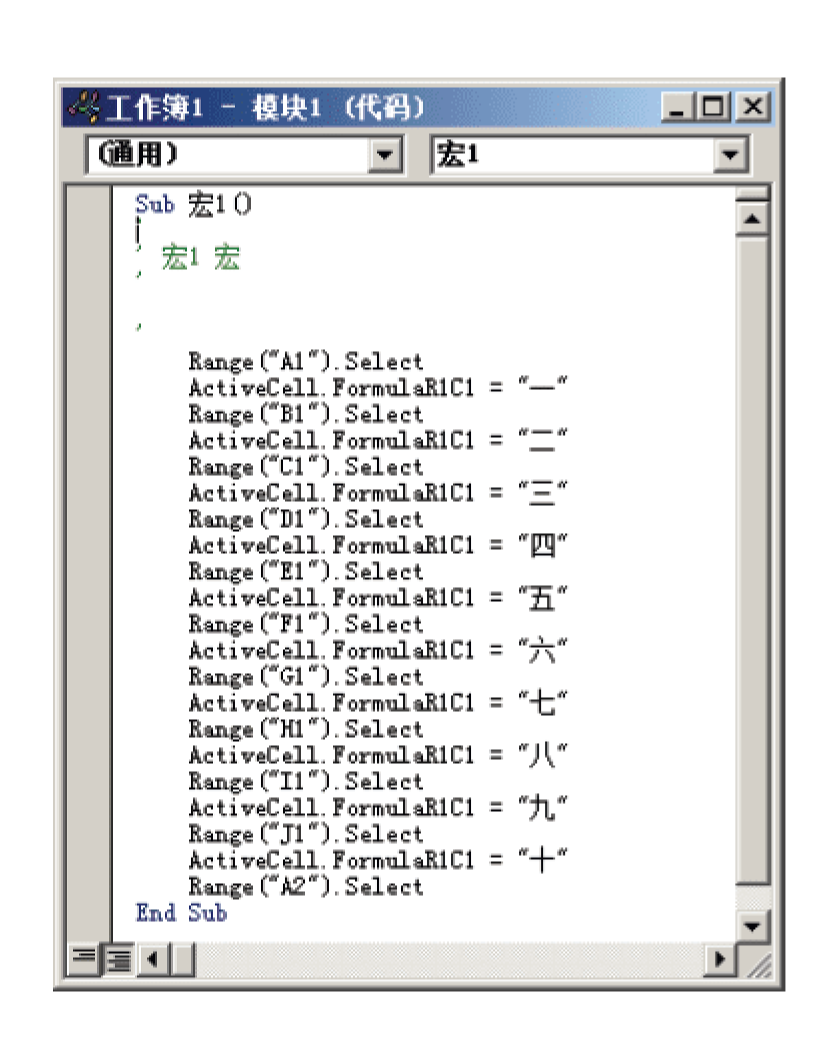


图 2.2　宏所对应的 VBA 程序代码

2.1.2　理解宏的功能

宏最主要的功能就是自动化频繁使用的命令。在电子表格的处理工作中，遇到需要重复做同一操作时，或者 Excel 2010 没有提供一个内置工具完成此任务时，就可以创建一个宏，方便以后工作使用。宏命令能够将工作表的任何部分工作自动化。

例如：① 自动化数据录入；② 检查选中的工作表区域里的重复值；③ 通过宏命令快速地将格式应用到多个工作表，并且可以结合不同的格式；④ 将图表创建和格式设置自动化；⑤ 设置页眉、页脚、页边距、打印区域以及选择特殊的打印选项等。

2.2　操作 Excel 中的宏

在 Excel 2010 中，宏的操作更加方便快捷。若要使用宏则必须先制作宏，然后才能使

用。所需用到的宏的操作主要包括录制宏、保存宏、执行宏、编辑宏、重命名宏、删除宏等。本节将介绍上述关于宏的基本操作。

2.2.1　录制宏

在录制宏的过程中，宏录制器会记录完成要录制的操作所需的一切步骤（有部分功能是无法完成录制的，例如，记录的步骤中不包括在功能区上导航的步骤）。因此在录制宏之前，要将所有需要录制的操作有一个详细的规划，同时要合理安排操作命令的顺序；否则录制的宏将包含大量的无关操作，最终影响宏的执行效率。

在开始录制宏之前有 3 项准备工作:

（1）确保“功能区”显示“开发工具”选项卡。

（2）将工作簿中宏的安全性级别设置为启用所有宏，其设置过程可参看“了解宏的安全性”。

（3）关闭所有包含宏或 VBA 代码的工作簿，以方便对录制宏的处理和定位。准备工作完成后，下面来制作一个简单的宏，用于设置文字格式。

下面以一个实例来介绍宏的录制过程。本实例录制一个宏，该宏能够在数据表中创建项目文字，同时设置文字的字体、字号和颜色等样式。具体操作步骤如下所示。

（1）启动 Excel 2010，创建一个名为“录制宏”的 Excel 文档。在“Sheet1”中选择第一个单元格，如图 2.3 所示。

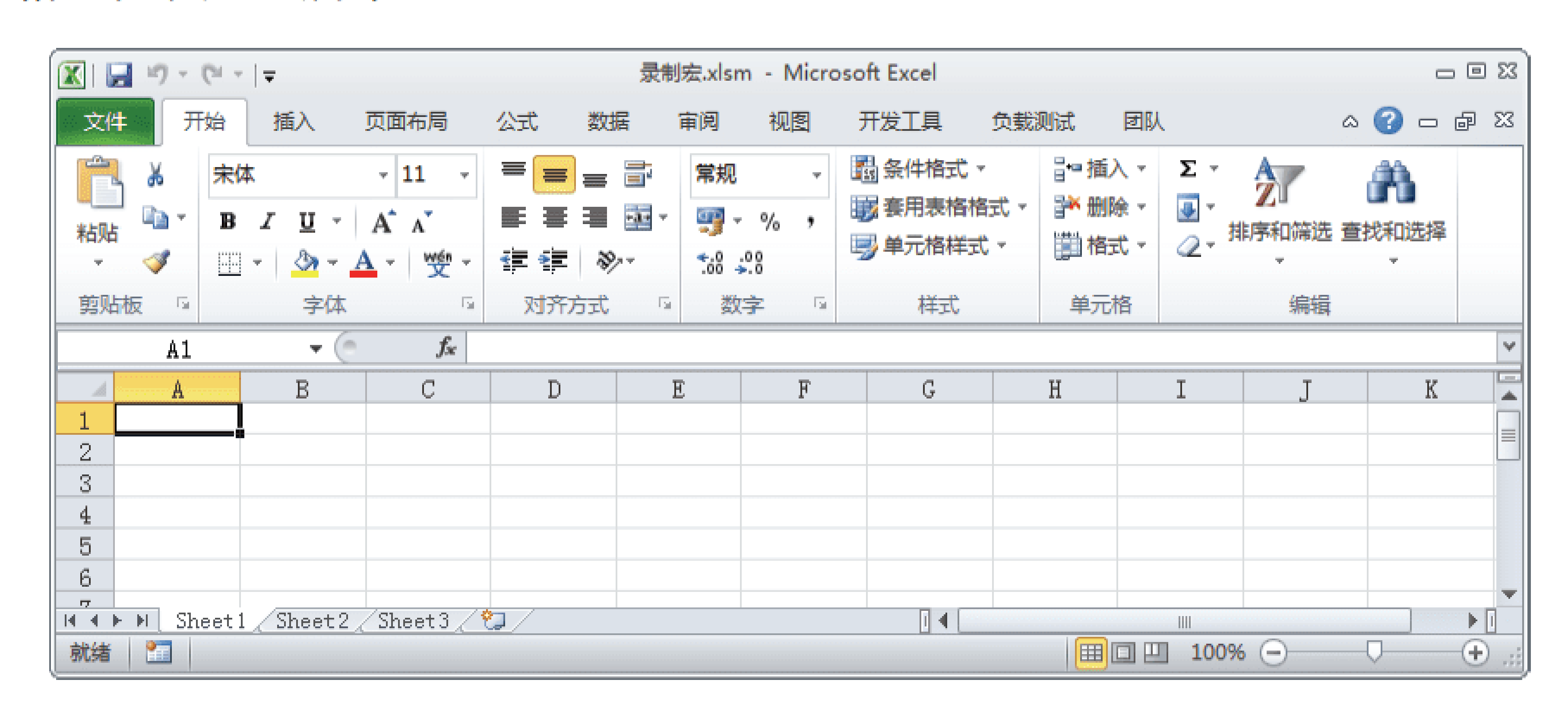


图 2.3　选择第一个单元格

（2）在 Excel 功能区上单击“视图”标签打开“视图”选项卡。单击“宏”按钮上的下三角按钮打开下拉列表。选择“录制宏”命令，如图 2.4 所示。

（3）在打开的“录制宏”对话框中修改宏的名称，同时为宏指定一个执行的快捷键。其他设置使用对话框中的默认值即可，如图 2.5 所示。

提示： 在“录制宏”对话框中默认的宏名为“宏 N”，其中 N 是从 1 开始的数字。为了便于宏的使用，应该给宏取一个既能够体现其功能又便于记忆的名称。为宏添加快捷键将便于宏的执行，如果需要创建更为复杂的组合键，可将输入点光标放置于“快捷键”输入框中后，按需要的快捷键即可。例如，使用 Ctrl+Shift+K 作为快捷键，可直接按 Ctrl+Shift+K 即可。

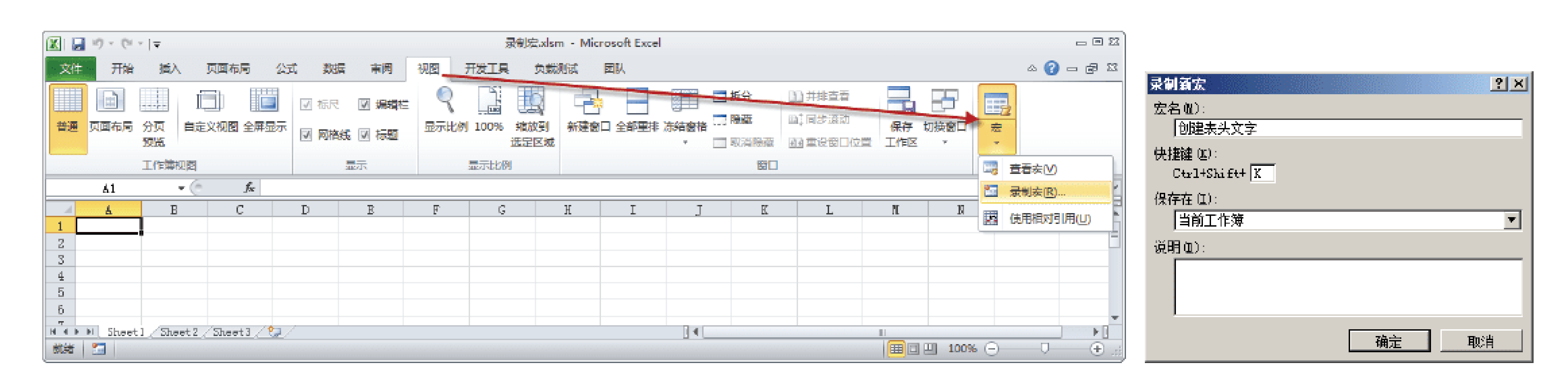
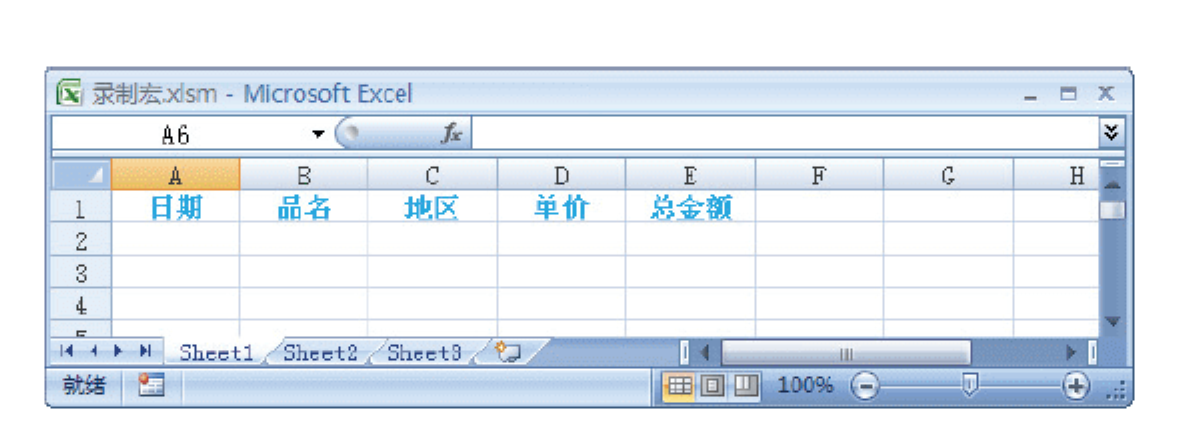


图 2.4　选择“录制宏”命令　　　　　图 2.5　“录制宏”对话框的设置

（4）单击“确定”按钮关闭“录制新宏”对话框，此时即进入宏录制状态，所有的操作将录制为宏。在 Excel 工作表中创建表头项目文字，并修改文字的样式，如图 2.6 所示。

（5）完成宏的录制后，在状态栏中单击“停止录制宏”按钮即可完成宏的录制，如图 2.7 所示。也可在“视图”选项卡的“宏”下拉列表中选择“停止录制宏”命令来停止宏的录制，如图 2.8 所示。



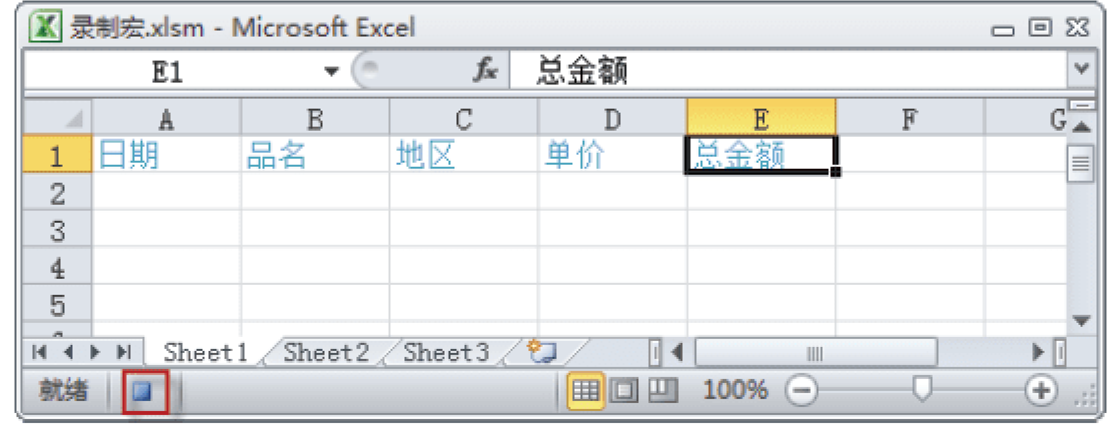


图 2.6　创建文字　　　　　图 2.7　“单击停止录制宏”按钮

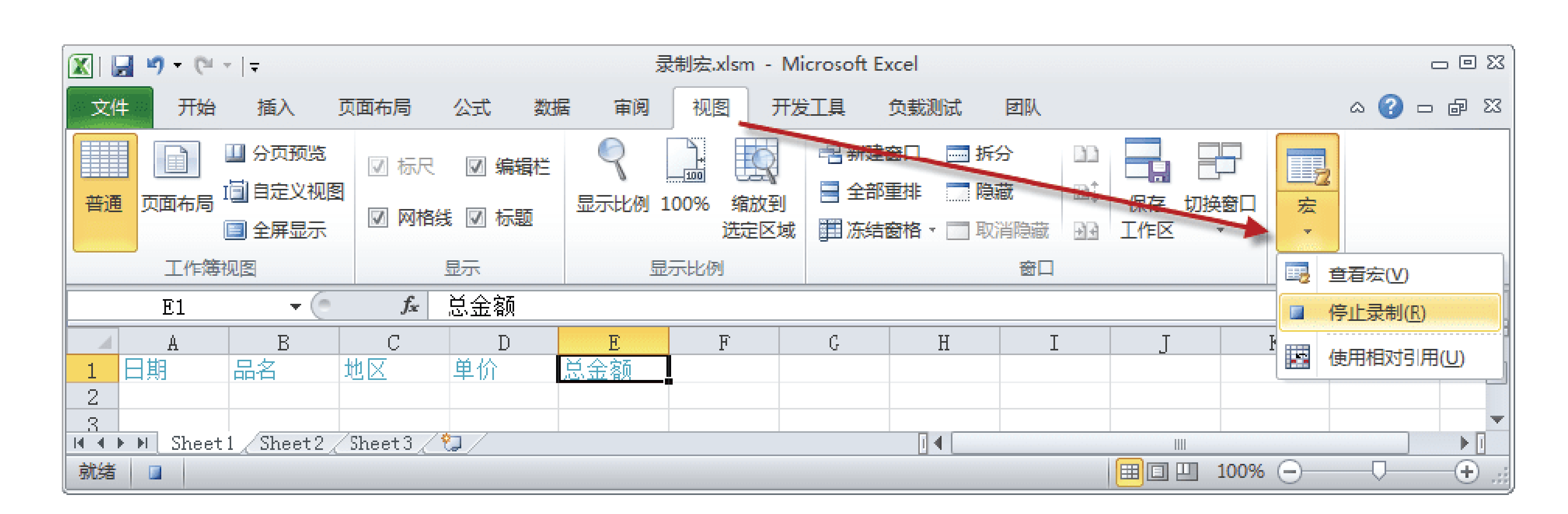


图 2.8　选择“停止录制”命令

提示：在录制宏的过程中，如果有对单元格的操作，则每次执行该宏时都将使用对应的单元格。这是因为宏记录的是对单元格的绝对引用。如果希望宏在执行时不考虑活动单元格的位置，可在录制宏前使“宏”下拉列表中的“使用相对引用”选项处于选择状态。此时录制的宏，单元格的引用将使用相对引用。

2.2.2　保存宏

宏录制完成后，需要将宏保存起来，供重复使用。选择“文件→保存”命令时，在弹出的保存对话框中含有“另存为”项，此处只是将新录制的宏保存在工作簿中，若需要以

后继续使用录制的宏，就需要保存该工作簿，单击“快速访问工具栏”上的“保存”按钮后，弹出“另存为”对话框，为文件命名，单击“保存”按钮即可。

注意： 对于含有宏的工作簿在保存时需要在“另存为”对话框中设置工作簿的“保存类型”，选择保存类型右边的下拉列表，选择“Excel 启用宏的工作簿”列表项，完成保存类型的设置，如图 2.9 所示。如果没有设定为此种类型，在单击“保存”按钮后，会弹出如图 2.10 所示对话框。

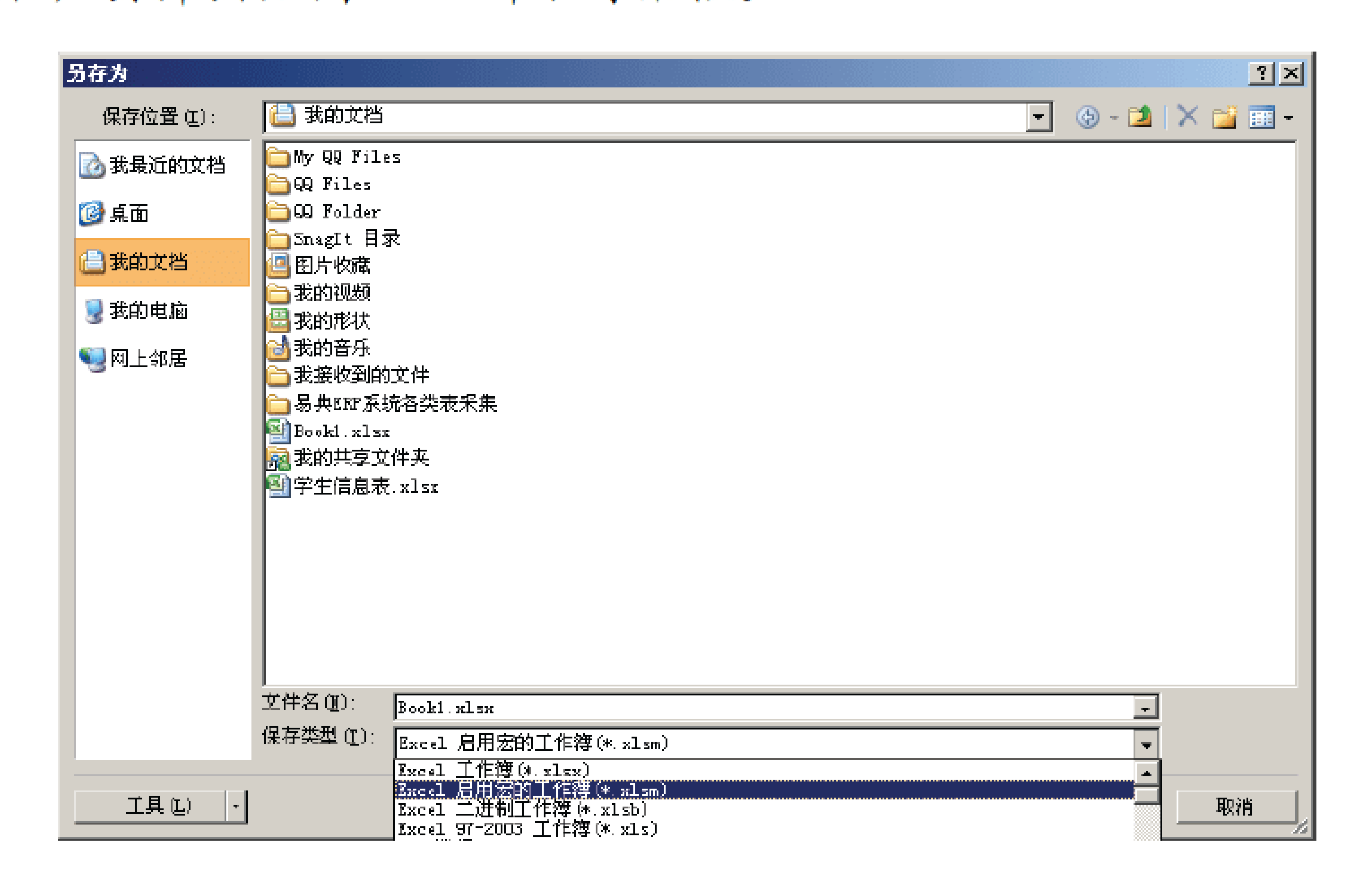


图 2.9　设置保存类型图

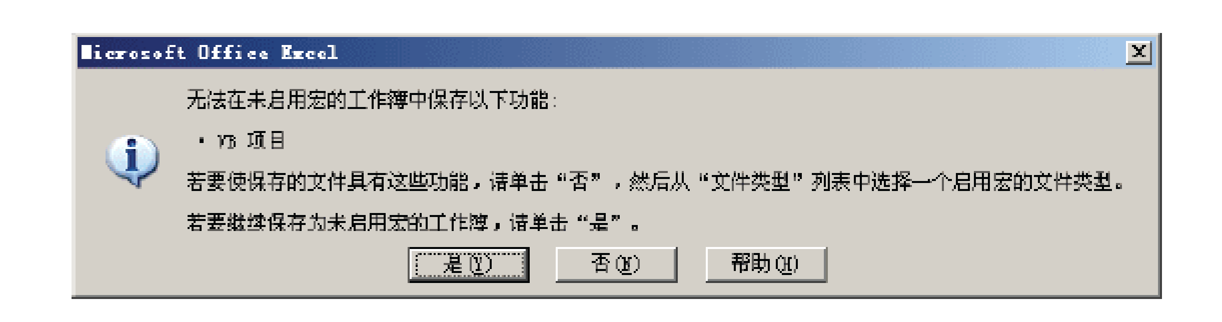


图 2.10　未设置保存类型时提示对话框

2.2.3　执行宏

录制宏是为了以后使用，使用宏就是要执行宏。Excel 2010 中有多种运行宏的方法。此处介绍 2 种常用的方法：

- ❑ 通过“开发工具”选项卡中的宏按钮来运行；
- ❑ 通过键盘快捷键来运行宏。

提示： 如果需要宏在工作表打开时自动运行，只需要在录制宏时将宏名设置为 Auto_Open 即可。Excel 中，Auto_Open 宏会在任何其他工作簿打开之前运行，但录制 Auto_Open 宏也将存在一些局限性。Auto_Open 宏中无法录制很多的操作，保存的 Auto_Open 宏在工作簿中已存在 Open 事件过程，Open 事件的 VBA 程序将覆盖 Auto_Open 宏中的操作。另外，如果一个工作簿是以 Open 方法打开的，则 Auto_Open 宏将被忽略。

此处以 2.2.1 节制作的宏为例介绍其运行方法。打开 2.2.1 节所保存的文档，选中 sheet2

工作表。

1. 通过宏按钮运行宏

（1）选择“开发工具”选项卡，在功能区“代码”组中单击“宏”按钮或直接按下 Alt+F8 快捷键，如图 2.11 所示，弹出“宏”对话框，如图 2.12 所示。

（2）单击对话框列表中相应的宏名。

（3）单击右侧“执行”按钮即可，宏运行效果如图 2.6 所示。

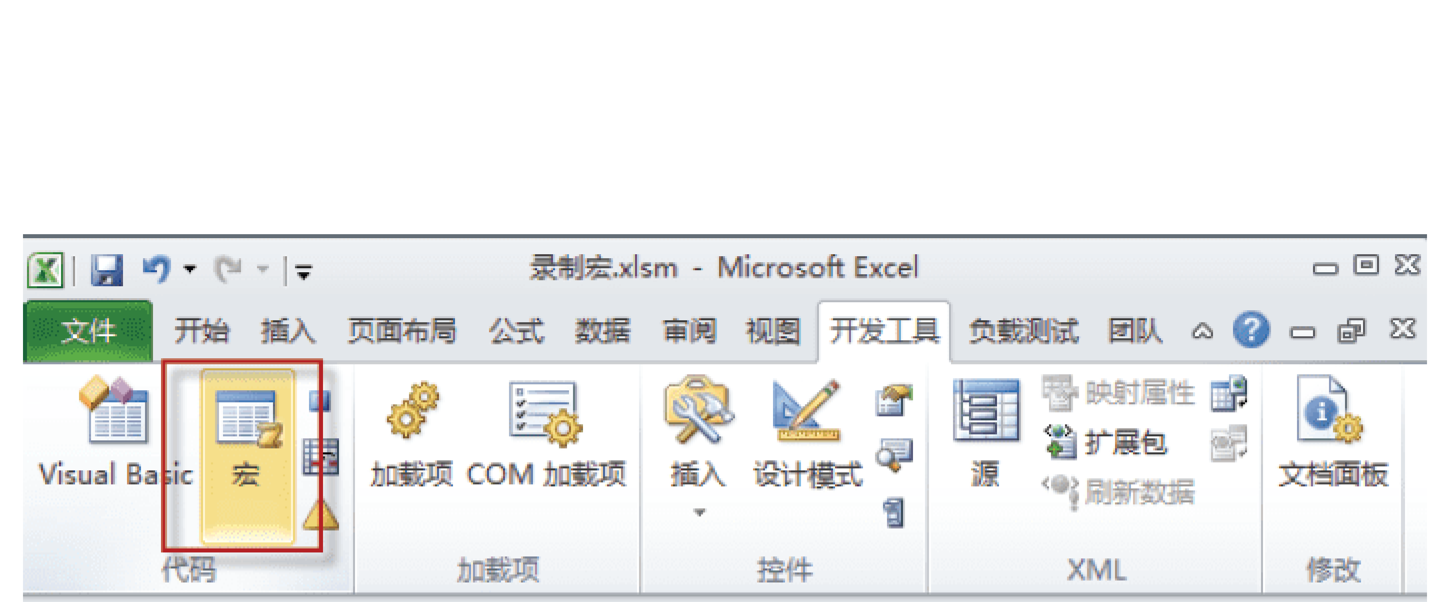


图 2.11　宏按钮图

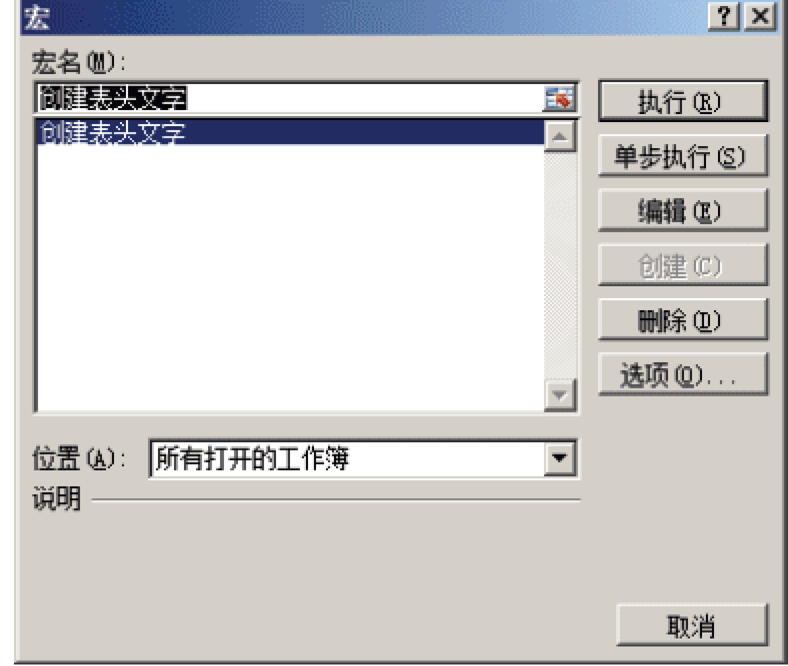


图 2.12　“宏”对话框

2. 通过快捷键运行宏

运用键盘快捷键的依据就是在录制宏时，需要为宏设置快捷键，如图 2.13 所示的对话框中有一项“快捷键”，在 2.2.1 节的宏中设置的键盘快捷键为 Ctrl+Shift+K。其操作方法是首先在工作表中，单击一个空白单元格，然后在键盘上同时按下 Ctrl+Shift+K 键，宏运行效果如图 2.14 所示。

注意：设置快捷时不要与 Excel 2010 中已有的快捷键重复，否则选择的宏不能正常执行。

2.2.4　编辑宏

宏记录器并不是万能的，是比较机械的，其记录的过程并不一定完全满足实际需求，此时就需要编辑已录制的宏。在 Excel 2010 中编辑宏，需要在 Visual Basic 编辑器中进行，其打开方法是在图 2.12 所示的对话框中单击“编辑”按钮后，弹出如图 2.13 所示的 Visual Basic 编辑器窗口。在 Visual Basic 编辑器窗口中可修改宏记录器记录的代码，关于宏代码的修改需要 VBA 编程的知识，在后续的章节会详细介绍 VBA 程序设计的知识。

本节仅介绍一些关于宏编辑的简单操作。在打开的 VBA 编辑器中更改宏名比较简单，在代码窗口中直接删除的第 1 行代码 sub 后的名字，重新输入新的宏名即可。当设置的快捷键与 Excel 2010 中已有的快捷键冲突时，可在图 2.12 所示的对话框中单击“选项”按钮后，弹出如图 2.14 所示的“宏选项”对话框，在快捷键下的文本框内输入字母键即可；若需要为宏追加说明，可直接在其说明下的编辑区中输入要追加的内容，单击“确定”按钮即可。

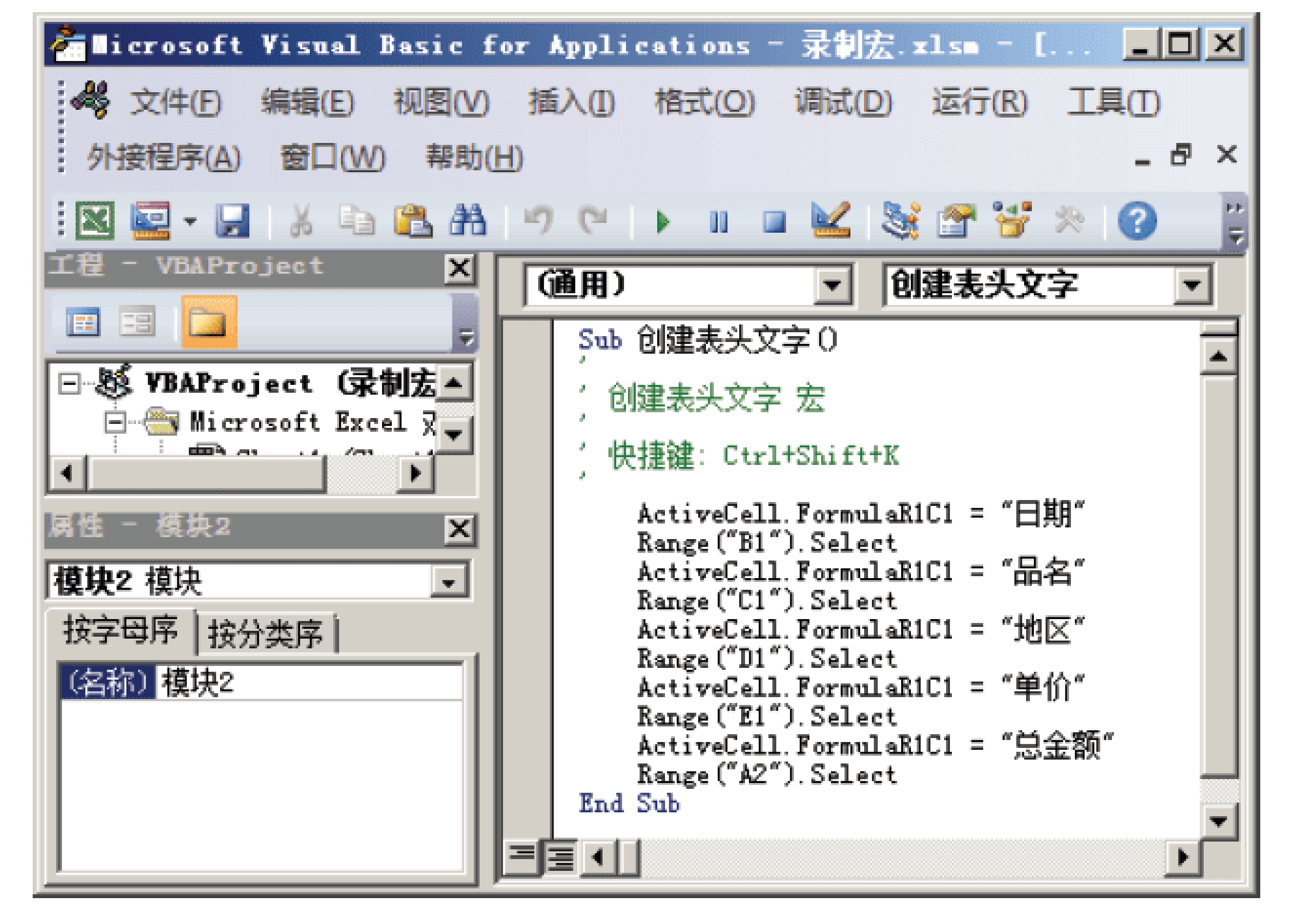


图 2.13　Visual Basic 编辑器窗口

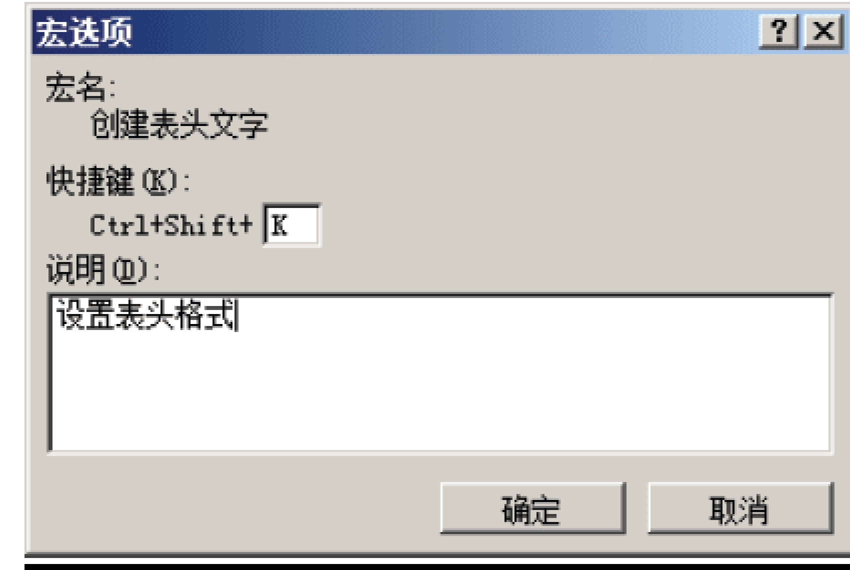


图 2.14　宏选项窗口

2.2.5　删除宏

随着实践工作的加深，可能不再需要先前录制的宏，可以将其从工作簿中删除。根据实际需要可将只将某个宏删除，也可以将包含该宏的模块删除；一个模块可以包含多个宏，当模块被删除时，包含在此模块中的所有宏都将会被删除。

1．通过宏列表删除

（1）在如图 2.15 所示的“宏选项”对话框中单击需要删除的宏。

（2）单击窗口右侧的“删除”按钮即可。

2．通过编辑器删除宏

在 Visual Basic 编辑器中进行操作，其操作步骤如下所述。选中需要删除的宏的所有代码，直接按键盘上的 Delete 键，然后关闭 Visual Basic 编辑器就可删除宏。删除模块时，首先打开 Visual Basic 编辑器中进行操作，其操作步骤如下所述。

（1）在工程资源管理窗口中单击相应模块。

（2）右击该模块，弹出如图 2.15 所示的快捷菜单。

（3）选择“移除模块”命令，此时会弹出如图 2.16 所示的提示窗口，依据实际需要单击相应的按钮，如果单击“否”按钮则将模块完全删除。

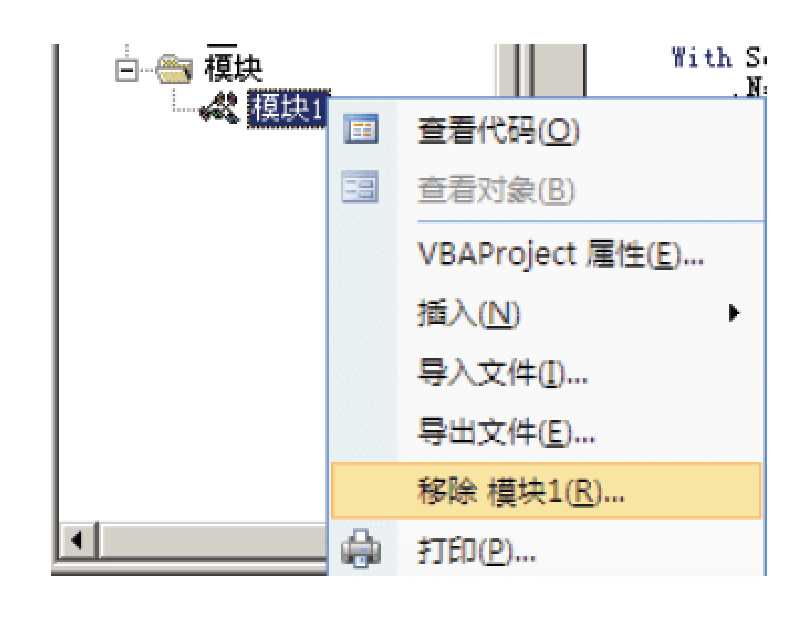


图 2.15　移除模块快捷菜单图



图 2.16　移除模块提示图

2.3　加　载　宏

Excel 2010 功能虽然强大，但也不可能解决所有的问题。对于 VBA 开发人员来说，Excel 提供了一个十分有用的功能，那就是创建加载宏功能。直接使用创建好的加载宏，能够节约开发时间，提高工作效率，无限扩展 Excel 功能。

2.3.1　在 Excel 中加载宏

所谓的加载宏，实际上是一类执行特定功能的程序。这些程序被保存后，用户可以在 Excel 启动后自动加载这些程序并使用它。使用加载宏可以为 Excel 增加新的命令和功能，通过加载它们，能够完成某些特殊领域或特殊数据处理要求下的任务。加载宏就如同很多应用程序中使用的插件，可以扩充 Excel 的处理能力，增强 Excel 的功能。在 Excel 2010 中加载宏一般分为以下 2 种类型。

- Excel 加载宏：这类加载宏包括 Excel 自带的加载宏和其他 Excel 加载宏。自带的加载宏在安装了 Excel 2010 后即可选择使用。其他 Excel 加载宏在下载安装后，可以像 Excel 自带加载宏那样使用。
- 自定义对象模型加载宏：这类加载宏是利用各种编程语言编写的 XLL 文件，用户需要掌握编程语言才能编写这类加载宏。

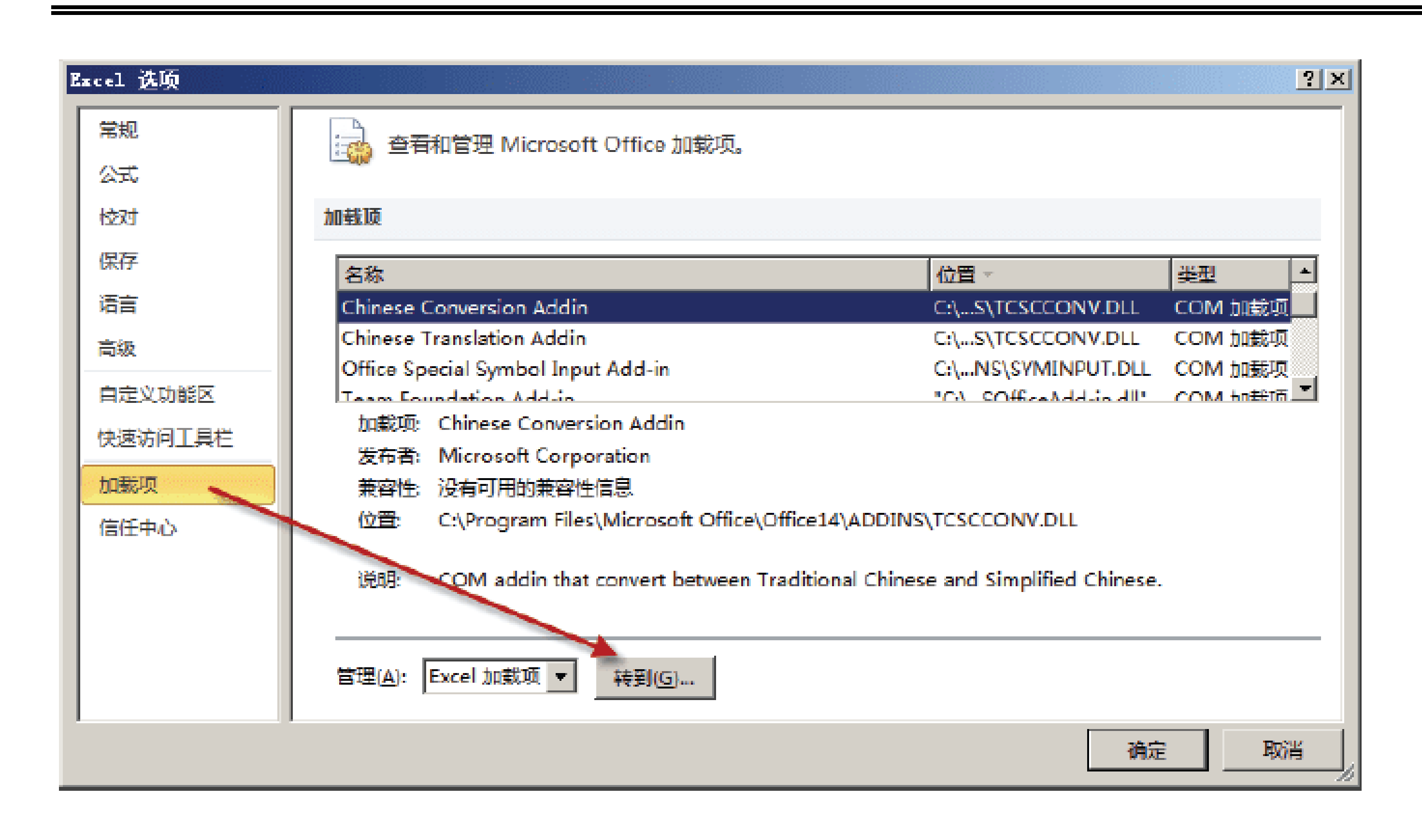
注意：在 Excel 2010 之前的版本中，大部分加载宏文件的扩展名均为“*.xla”，Excel 2010 中加载宏的扩展名为“*.xlam”。过去版本中的大多数加载宏在 Excel 2010 中都可以使用。在 Excel 2010，加载宏存放路径一般在“C:\Documents and Settings\用户名\Application Data\Microsoft\AddIns”文件夹中。

在安装 Excel 2010 后，Excel 2010 自带的加载宏会安装在计算机中。这些加载宏文件被安装在 Office 安装目录下 office12 文件夹的 Libray 文件夹中。下面介绍使用 Excel 2010 自带的加载宏的方法。

（1）选择“文件”选项卡，在打开的菜单中选择“选项”命令项，打开“Excel 选项”对话框。在对话框中左侧窗格中单击“加载项”选项，此时即可看到所有 Excel 加载项列表。在对话框中单击“转到”按钮，如图 2.17 所示。

（2）此时可以打开“加载宏”对话框。在对话框中选中需要加载的加载宏，单击“确定”按钮完成加载宏的加载，如图 2.18 所示。此时，在功能区中将能找到添加的加载项按钮，如图 2.19 所示。单击该按钮即可使用该加载宏来进行数据处理。

注意：加载的加载宏会根据其不同的用途放置在不同的选项卡中。如：“分析工具库”加载宏和“分析工具库－VBA”加载宏被添加到“数据”选项卡中，“查询向导”加载宏被添加到“公式”选项卡中。



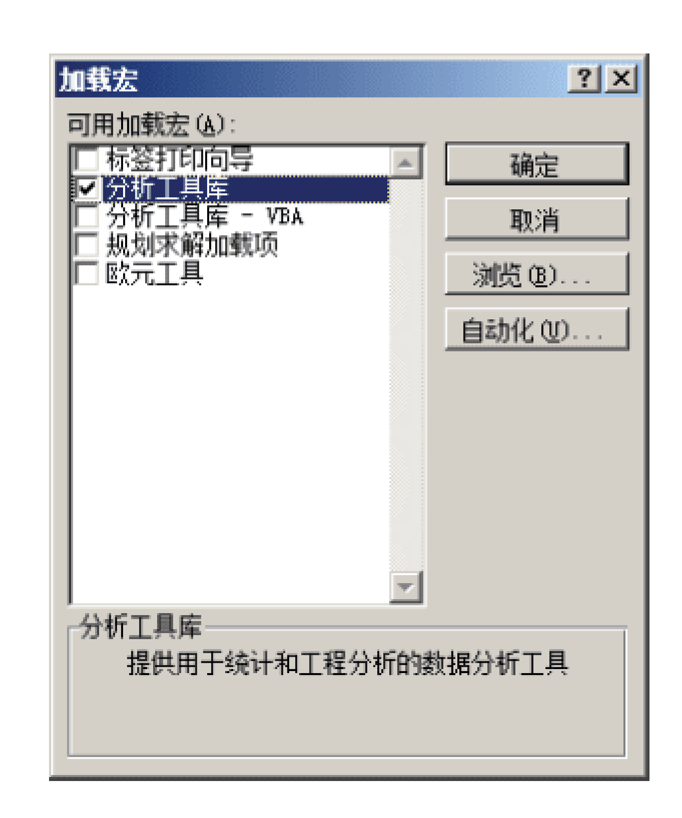


图 2.17　“Excel 选项”对话框中的加载项　　　　图 2.18　在“加载宏”对话框中选择加载宏

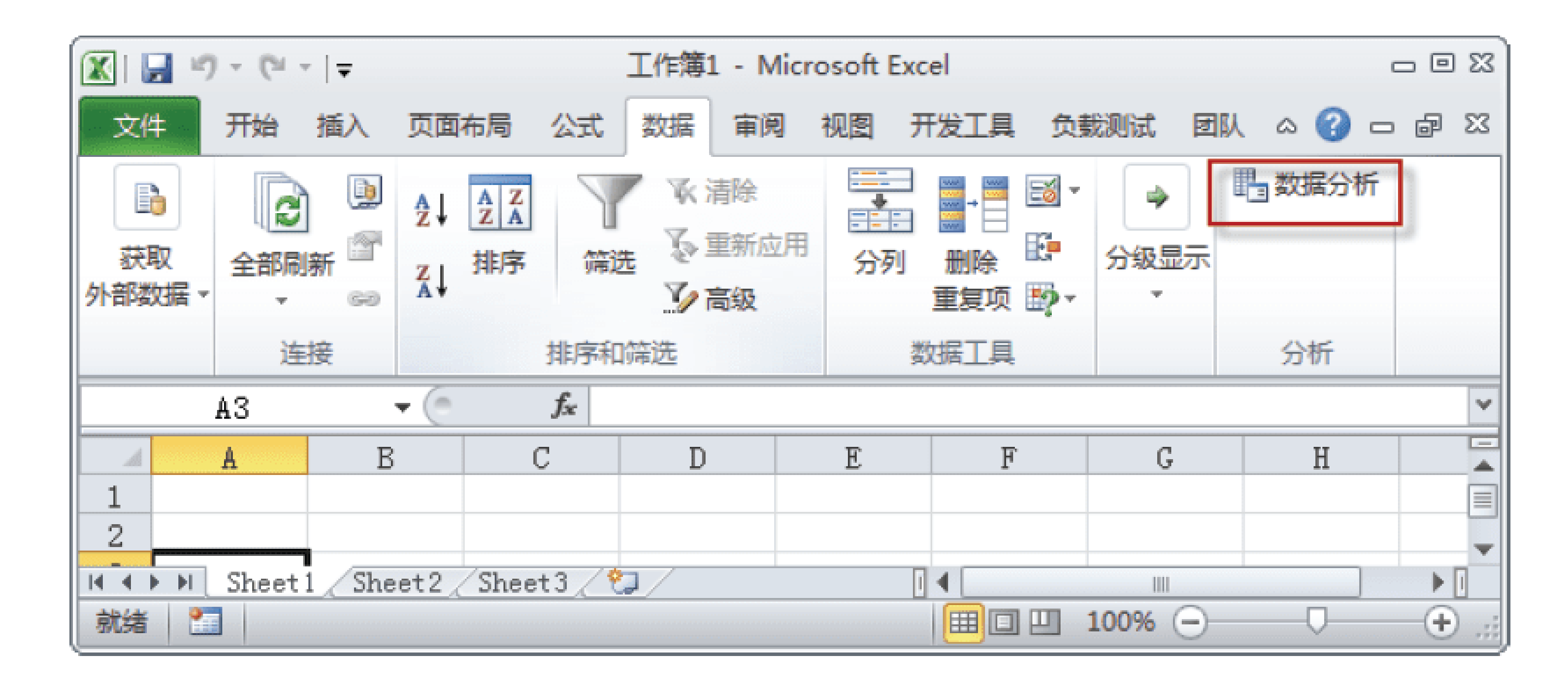


图 2.19　加载项添加到功能区中

2.3.2　在 Excel 中卸载加载宏

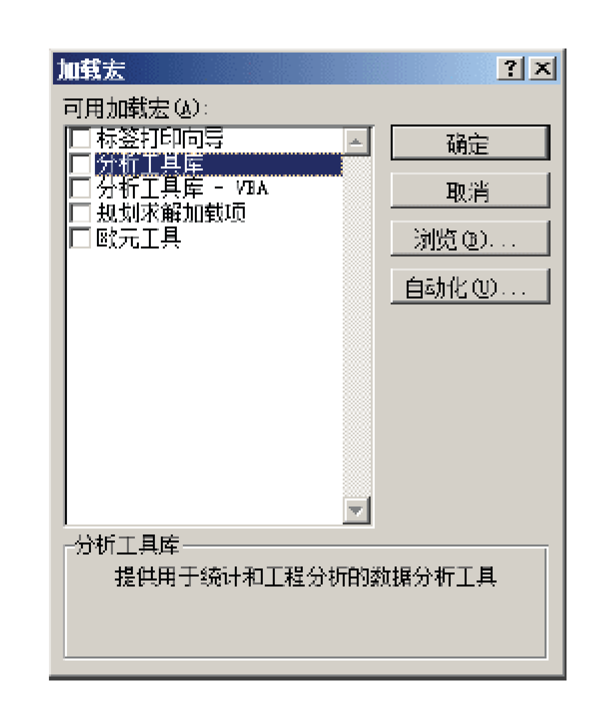


图 2.20　取消要卸载的加载宏的勾选

将加载宏载入 Excel 后，每次启动 Excel 时都将被自动载入。添加到功能区的加载宏按钮，在不需要时可以将其卸载。卸载的方法如下所示。

（1）打开“加载宏”对话框。单击所要取消的加载宏，使其前面的复选框前“√”去掉，单击“确定”按钮关闭对话框，如图 2.20 所示。

（2）此时功能区中原有的加载宏按钮将被删除，加载的加载宏被卸载，如图 2.21 所示。

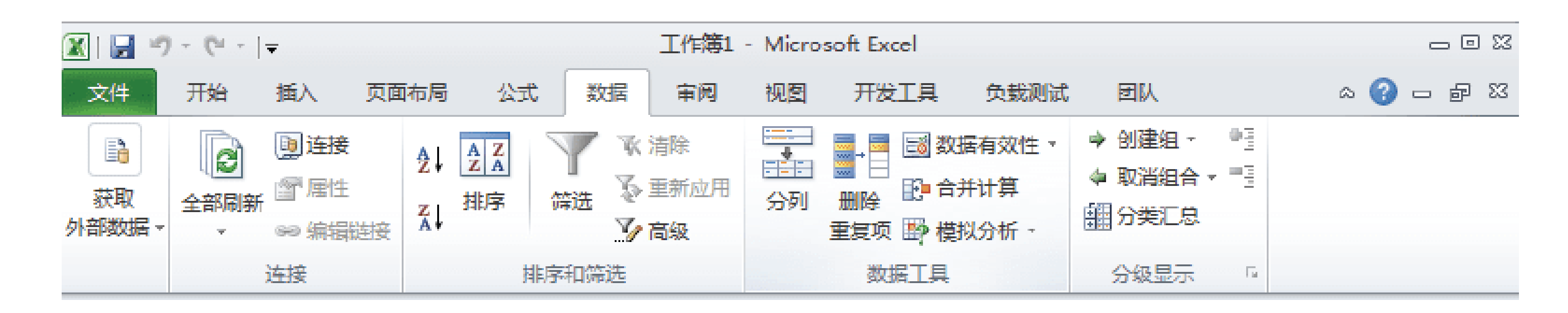


图 2.21　功能区中的加载宏按钮被删除

提示： 卸载加载宏后，加载宏选项在“加载宏”列表中仍然存在。如果希望将其从列表中删除，可以从磁盘上删除该加载宏文件。当再次在“加载宏”列表中选择该项

时，Excel 会提示其不存在，是否删除。选择删除后，“加载宏”列表中将不会再有这个选项了。

2.3.3　在 Excel 中保存加载宏

自己录制的加载宏，不仅可以用于当前工作簿，也可以用于其他的工作簿。如果需要在本地计算机的其他工作簿中使用宏，可以将宏保存在“个人宏工作簿”中。而如果需要在其他的计算机中使用这些宏，可以使用“加载宏”来实现。要使用自己的加载宏，必须将带有宏的 Excel 文档保存为加载宏文件，下面介绍保存加载宏的方法。

（1）打开含有宏的 Excel 2010 文件。选择“文件→另存为”命令，打开“另存为”对话框。在“保存类型”下拉列表中将文档的保存类型设置为“Excel 加载宏（*.xlam）”，如图 2.22 所示。

（2）此时，“另存为”对话框的保存文字将自动跳到 Excel 加载宏的默认保存文件夹中。如果希望将加载宏保存在该文件夹中，只需要单击“确定”按钮即可，如图 2.23 所示。

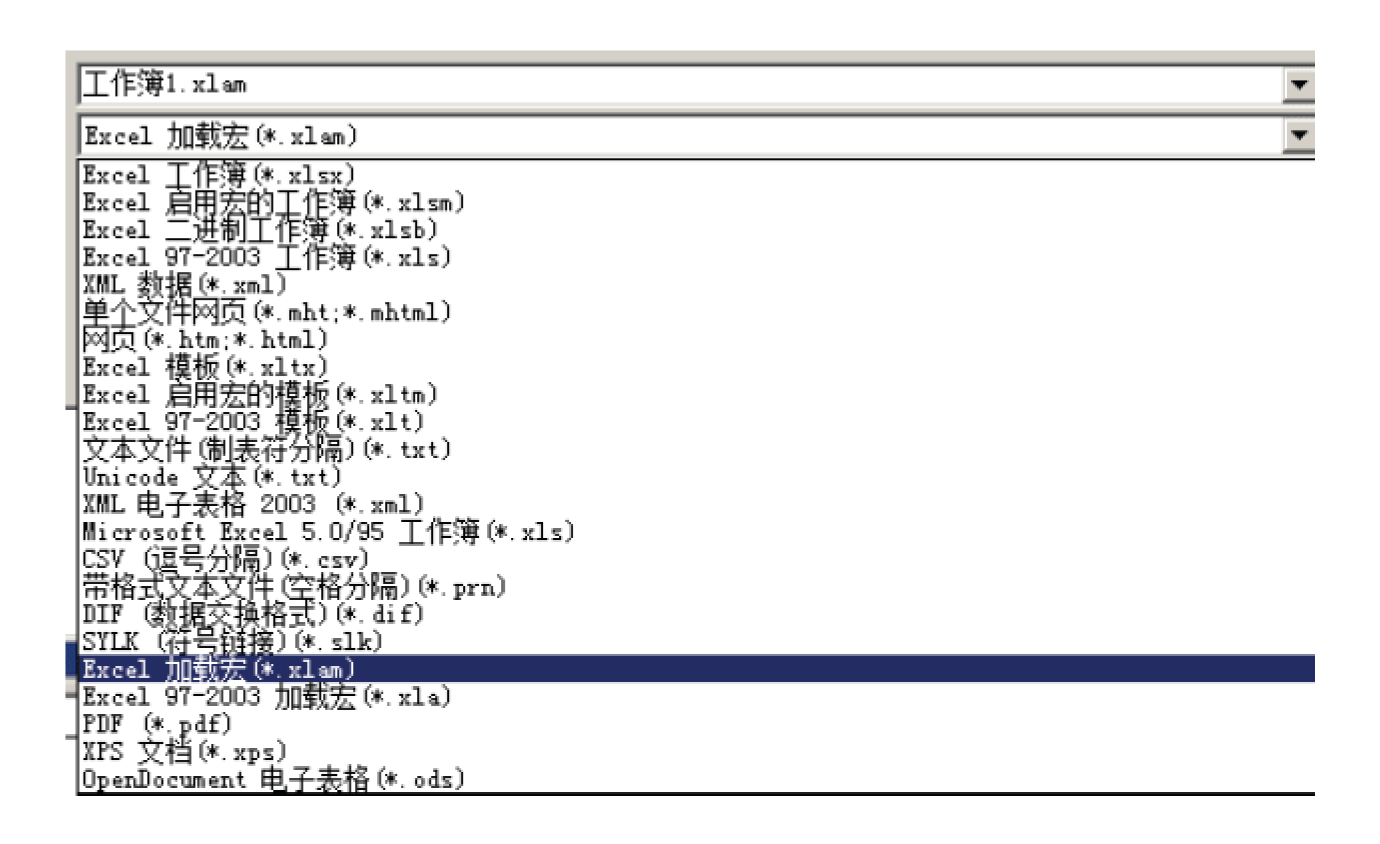


图 2.22　设置文档保存类型

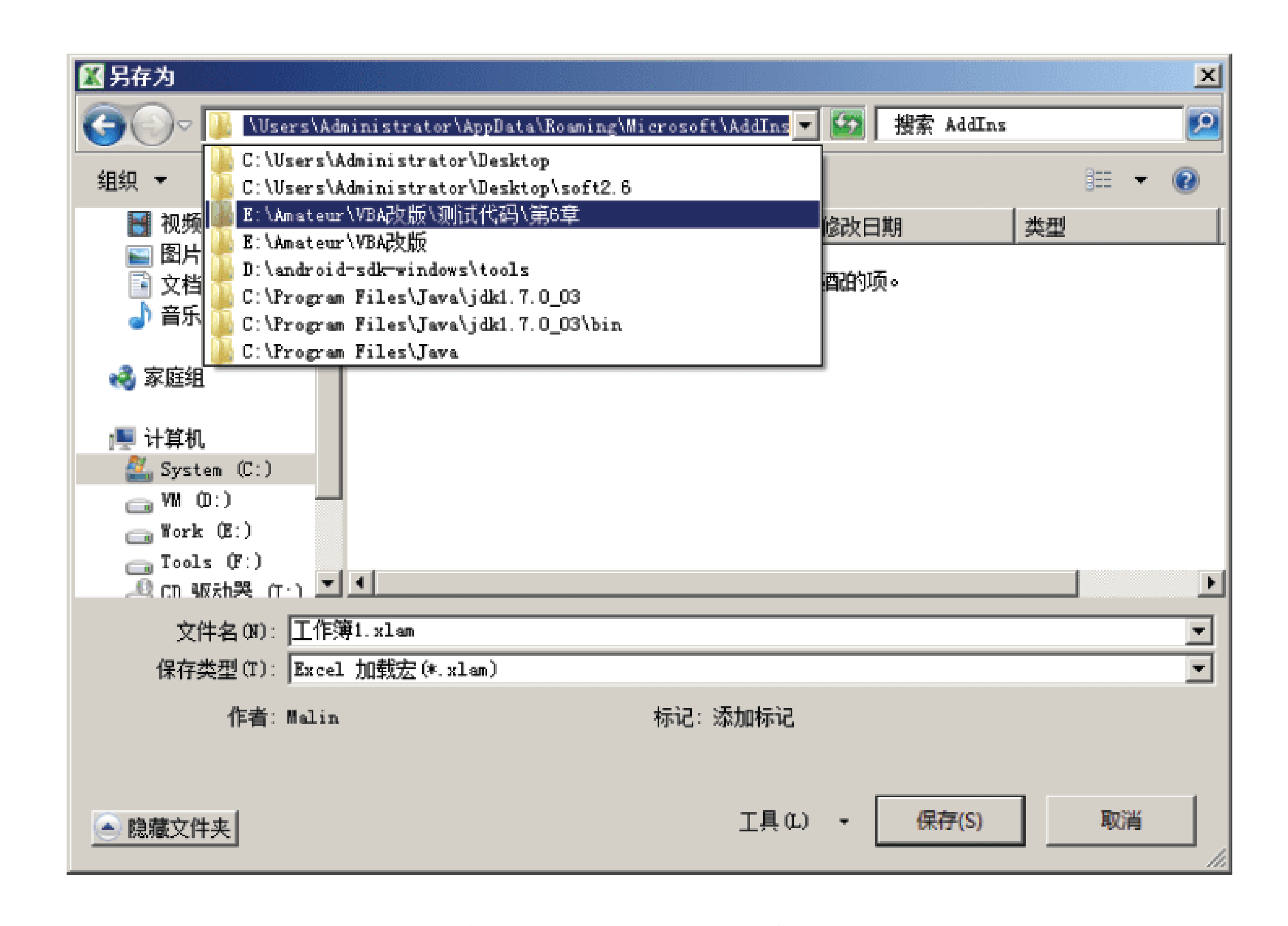


图 2.23　单击“确定”按钮保存加载宏

注意：Excel 加载宏在保存时实际上是一个文件，可以根据需要保存在磁盘的任何位置。具体的保存位置，可以在“另存为”对话框的“保存位置”下拉列表中根据需要选择。

2.3.4　Excel 中的其他加载宏

在 Excel 中，可以以加载宏的形式加载用户自己创建的宏来获得 Excel 不具有的额外功能。要使用用户创建的加载宏，首先需要在工作簿中加载这些宏。本节将介绍加载用户创建的加载宏的方法。

（1）打开“加载宏”对话框，单击其中的“浏览”按钮，如图 2.24 所示。

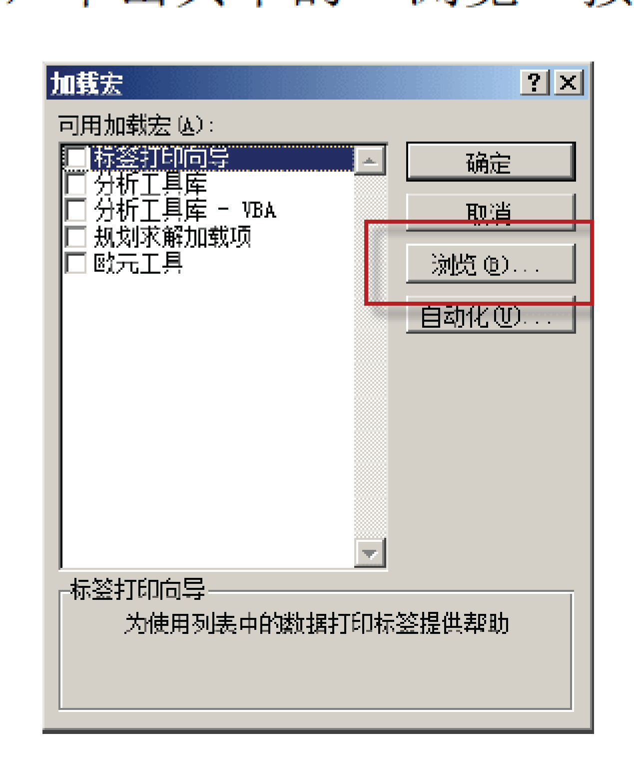


图 2.24　单击“加载宏”对话框中的“浏览”按钮

（2）在打开的“浏览”对话框中选择需要加载的加载宏文件，单击“确定”按钮，如图 2.25 所示。此时，该加载宏被添加到“加载宏”对话框中。选中添加的加载宏后单击“确定”按钮，如图 2.26 所示。至此，该加载宏被添加到 Excel 文档中。

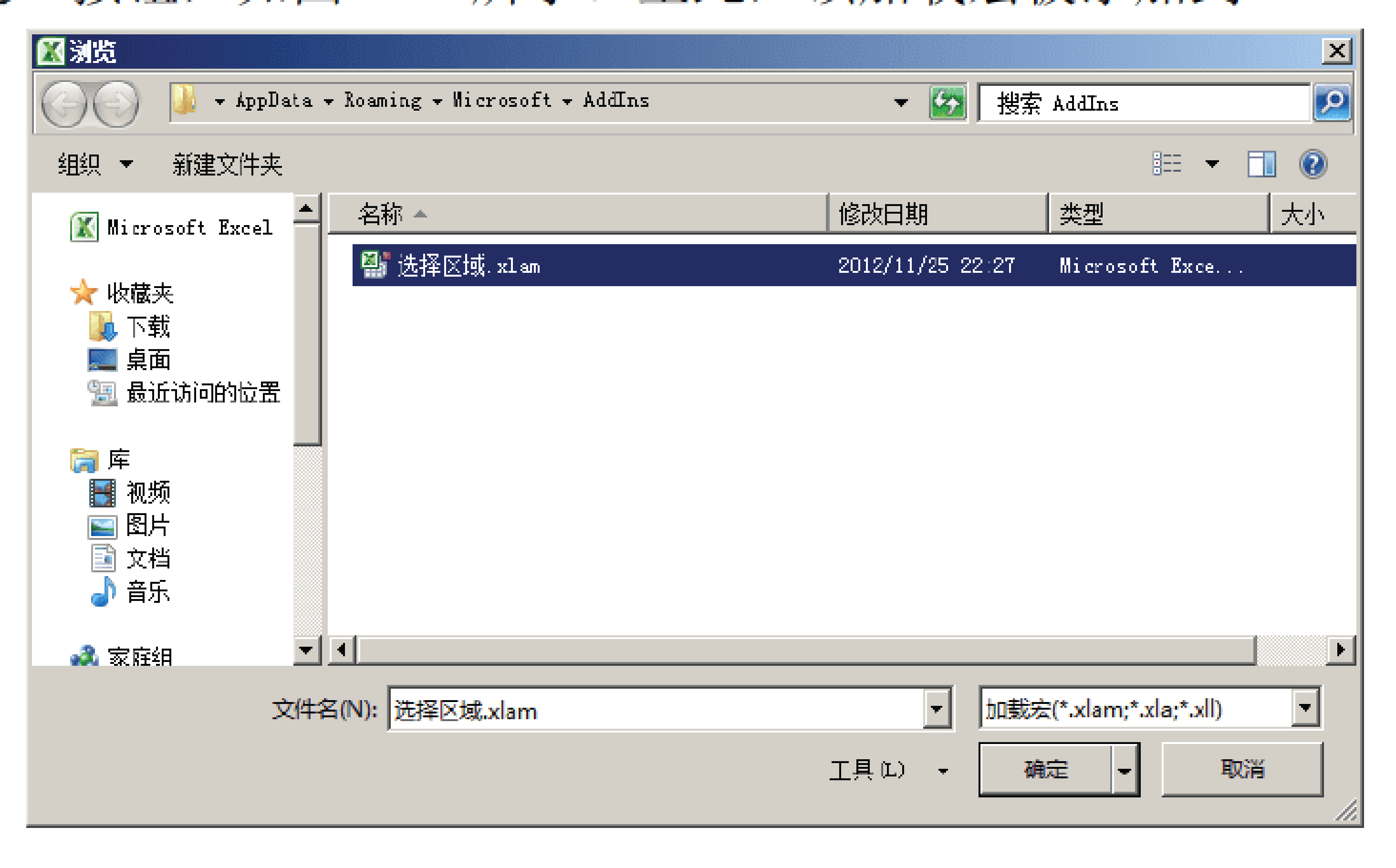


图 2.25　选择需要加载的加载宏

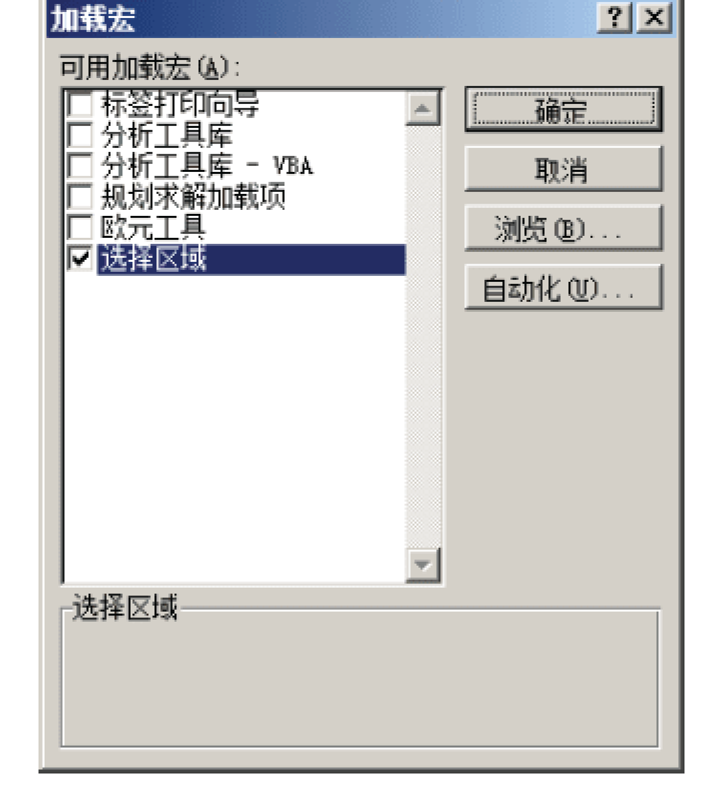


图 2.26　添加加载宏

注意：对于某些非 Excel 自带的加载宏，在加载后可能无法像 Excel 自带的加载宏那样在功能区中找到其按钮。此时，可以通过在 Visual Basic 编辑器中运行宏的方法来运行此类加载宏。

2.4　宏的安全性

宏的作用是使任务自动化，熟练的开发人员能够使用代码编写功能强大的 VBA 宏，这些代码能够在计算机上自动完成复杂的工作，对数据文件及系统进行操作。因此，宏的存在也就会带来潜在的安全隐患。自从 Office 开始支持宏以来，宏病毒就应运而生，并且曾经猖獗一时，造成了巨大的损失。Excel 2010 提供了比以前版本更加细致的宏安全性设置，正确地使用能够在大多数情况下杜绝宏病毒所带来的危害。本节将介绍 Excel 2010 宏安全性设置的方法。

2.4.1　通过信任中心设置宏的安全

Excel 2010 允许用户通过选择宏安全设置来实现对包含宏的 Excel 文档中宏的行为的控制。设置宏的安全性，一般采用下面的操作步骤。

（1）启动 Excel 2010，选择“文件→选项”命令，打开“Excel 选项”对话框。在左侧窗格中选择“信任中心”选项，在右侧窗格中单击“信任中心设置”按钮，如图 2.27 所示。

（2）在打开的“信任中心”对话框中选择“宏设置”选项，在右侧的窗格中即可对文档中宏的运行进行设置，如图 2.28 所示。单击“确定”按钮关闭“信任中心”对话框和“Excel 选项”对话框，即可完成宏安全性的设置。

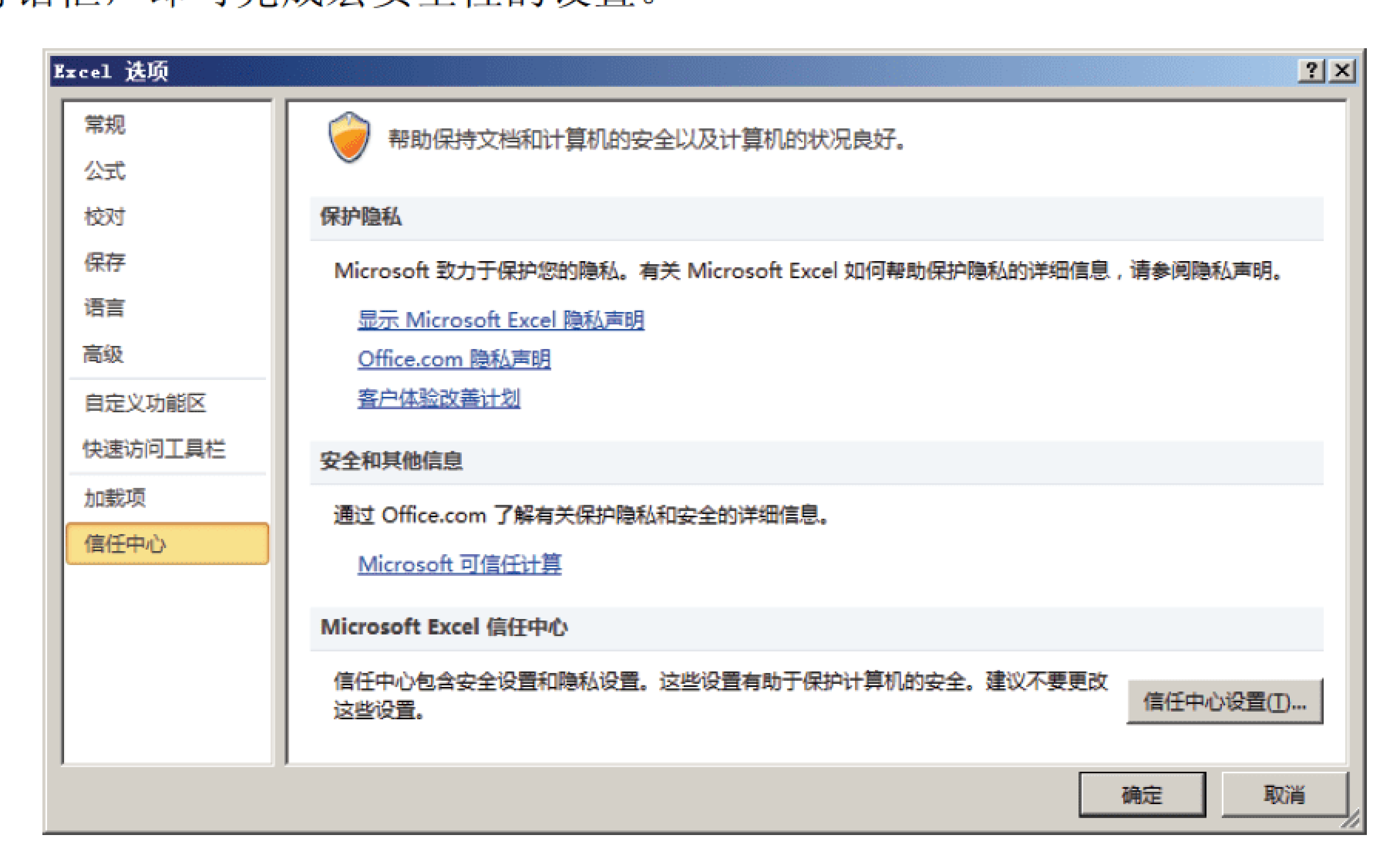


图 2.27　单击“信任中心设置”按钮

提示： 选中图 2.28 中“禁用所有的宏，并且不通知”单选按钮，文档中所有的宏及有关的安全警报将全部禁用，该选项的安全级别也是最高的。如果选中“禁用所有的宏，并发出通知”单选按钮，宏的运行将被禁止，但 Excel 会给出安全警报。此时，用户可以根据需要选择是否启用宏。选中“禁用无数字签署的所有宏”单选按钮，当宏已由受信任的发行者进行了数字签名后，如果是信任的发行者，则宏

可以直接运行，否则将发出安全警报。如果选中“启用所有宏（不推荐，可能会运行潜在危险的代码）”单选按钮，所有的宏都将可以运行。

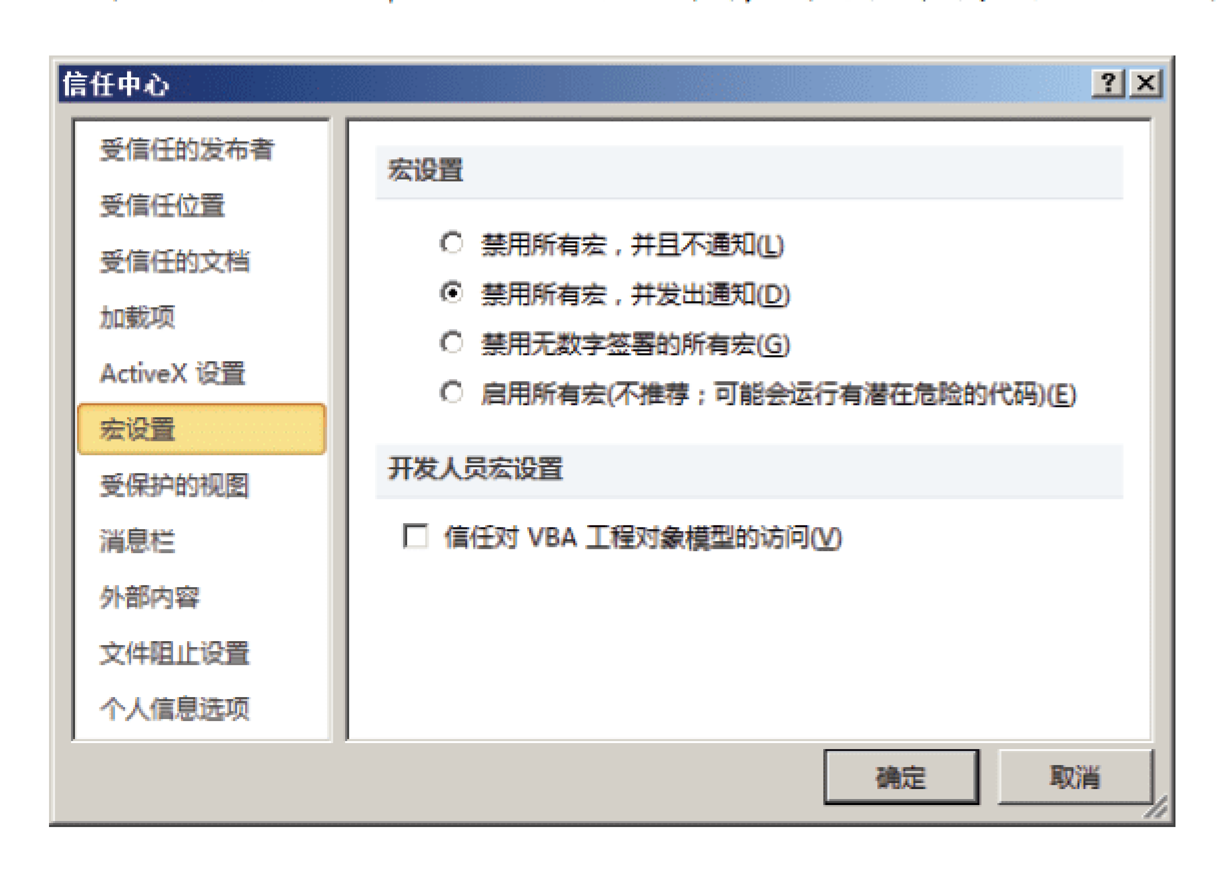


图 2.28　“信任中心”对话框中的宏安全设置

2.4.2　通过信任中心启用被禁的宏

在“信任中心”将宏的运行设置为“禁用宏，并发出通知”后，Excel 会给出安全警报。此时用户可以根据需要来选择是否运行宏。下面介绍具体的操作方法。

（1）启动一个带有宏的 Excel 文档，此时在功能区的下方将给出安全警告，如图 2.29 所示。

（2）单击“安全警告”栏上的“选项”按钮将打开“信任中心”对话框，如图 2.30 所示。如果允许文档中的宏运行，选中对话框中的“启用此内容”单选按钮选择该项，单击“确定”按钮关闭该对话框。此时，文档中的被禁用的宏就可以正常运行了。

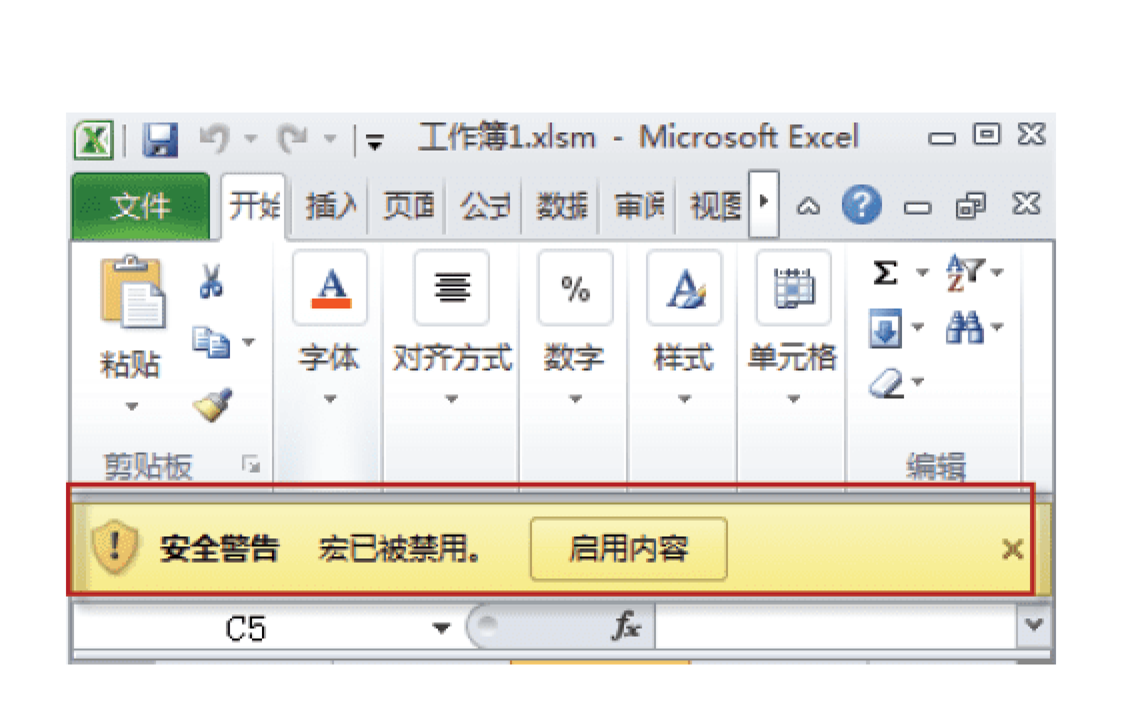


图 2.29　宏安全警告

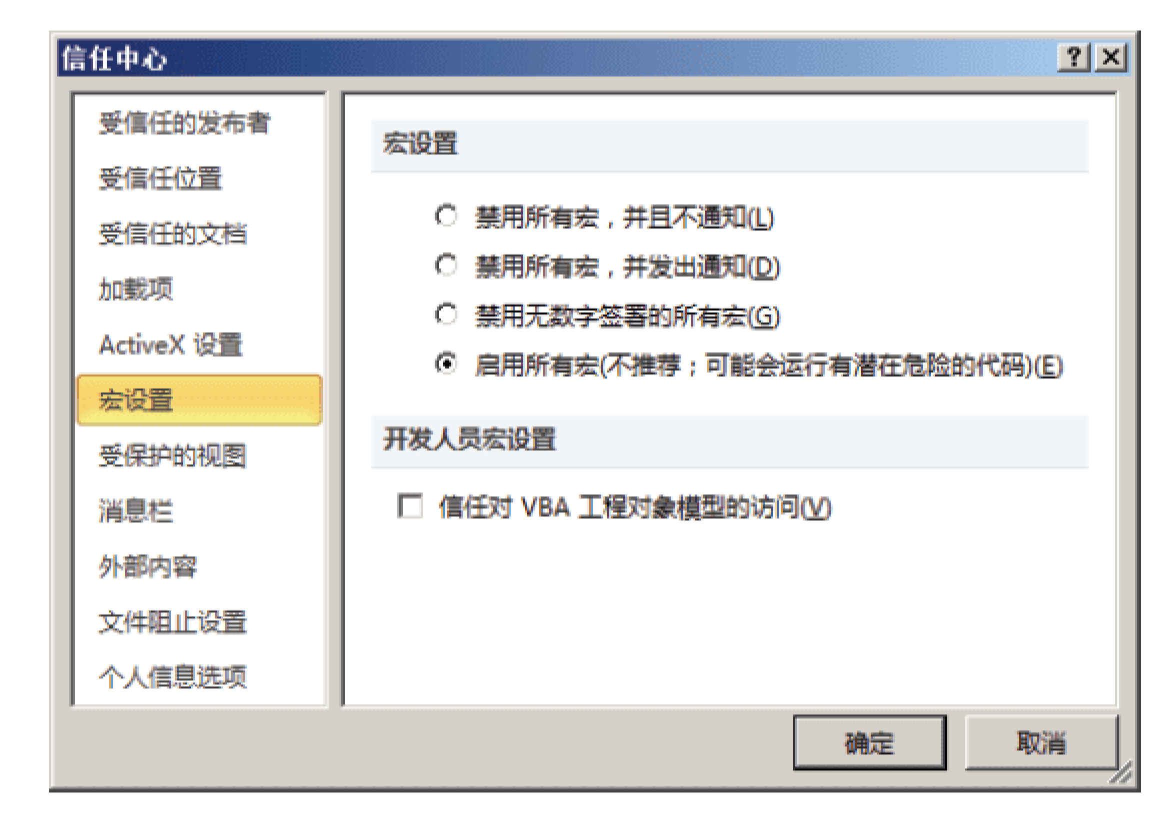


图 2.30　启用被禁用的宏

2.5　宏的数字签名

所谓的数字签名，是在宏或电子文档上的一种电子的、加密的安全验证印记。数字签

名的目的和作用与传统的纸质文档的签名警报相同，但它是一种使用计算机加密来验证的数字信息。数字签名有助于确保数字信息的真实性、完整性和不可否认性，是一种文档保护的重要方法。本节将主要介绍 Excel 2010 中数字签名的使用方法。

2.5.1　添加数字签名

在 Excel 2010 中，可以向文档中添加可见的签名行以获取一个或多个数字签名。在向文档添加了这种数字签名后，文档属性变成只读，这就像给文档添上了一把锁那样。这样，可以防止文档被随意修改，保证文档的可靠。下面介绍为文档添加签名行的操作方法。

（1）启动 Excel 2010，选择需要添加签名行的单元格。选择“插入”选项卡，单击“签名行”按钮。在打开的菜单中选择“Microsoft Office 签名行”命令，如图 2.31 所示。

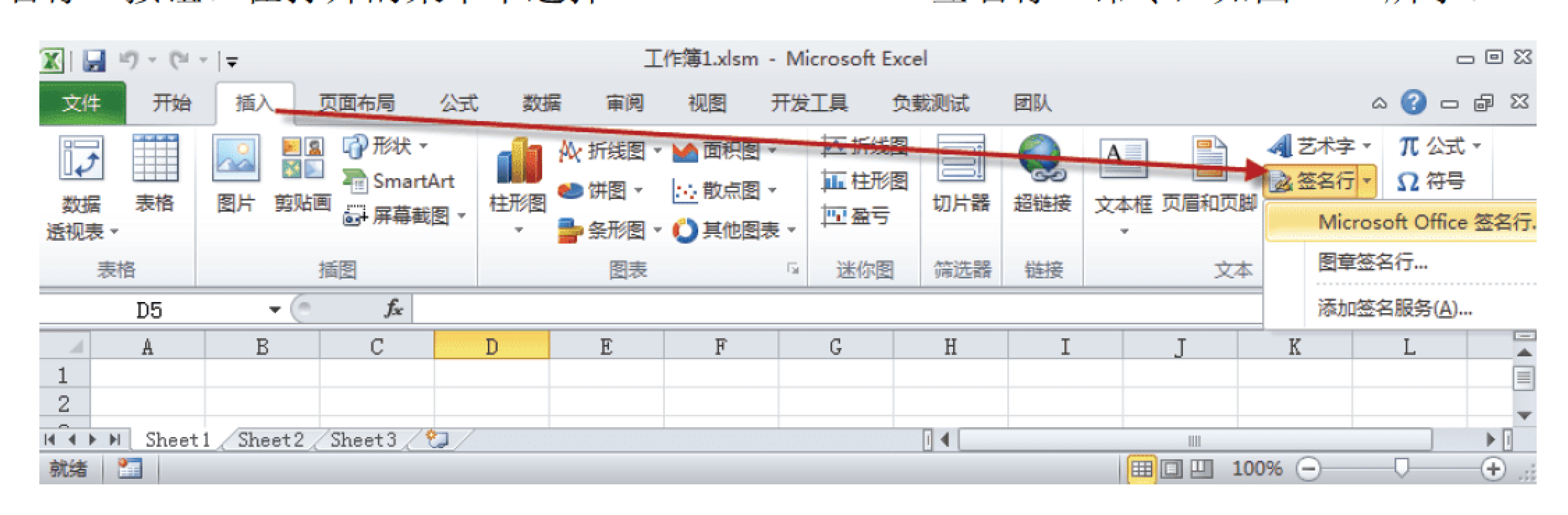


图 2.31　选择“Microsoft Office 签名行”命令

此时，Excel 2010 给出 Microsoft Office Excel 提示对话框，如图 2.32 所示。单击“确定”按钮关闭该对话框。

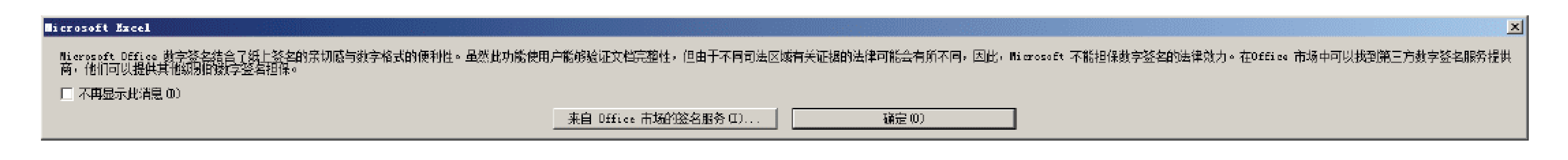


图 2.32　Microsoft Office Excel 提示对话框

（2）在打开的“签名设置”对话框中输入签署人员的相关信息，如图 2.33 所示。

提示： 在“签名设置”对话框中，“建议的签名人”文本框用于输入签署人姓名，“建议的签名人职务”文本框用于输入签署人组织职务，“建议的签名人电子邮件地址”文本框可输入签署人邮箱地址。如果要向签署人提供说明，则可以在“签名人说明”文本框中输入。

（3）完成信息添加后，单击“确定”按钮关闭“签名设置”对话框，在选择的单元格旁将出现设置的签名信息，如图 2.34 所示。

（4）在设置好的签名行上双击。如果是第一次进行数字签名，则会打开“获取数字标识”对话框，根据需要在对话框中进行设置，如图 2.35 所示。

（5）关闭“获取数字标识”对话框后即可打开“签名”对话框。在对话框中的“X”旁的文本框中输入姓名后单击“签名”按钮，如图 2.36 所示。此时，Excel 会给出“签名

确认”对话框，如图 2.37 所示。直接单击“确定”按钮关闭该对话框即可。

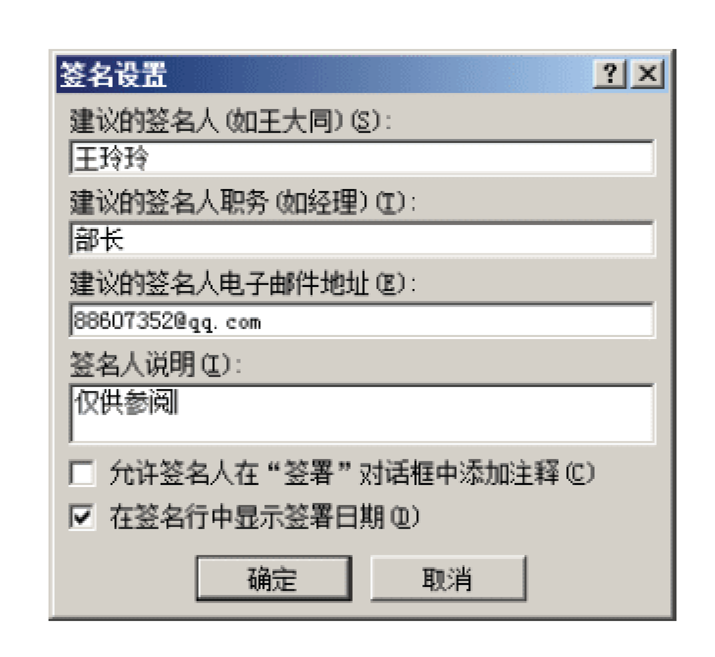


图 2.33 “签名设置”对话框

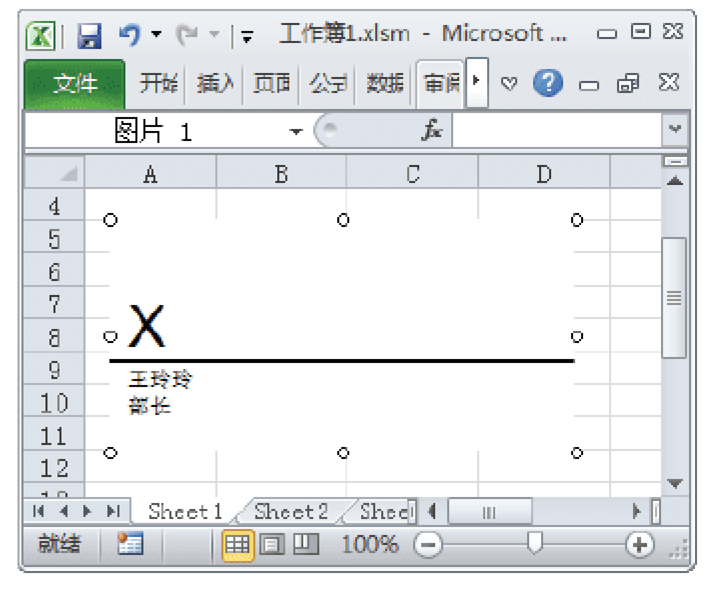


图 2.34 出现签名信息

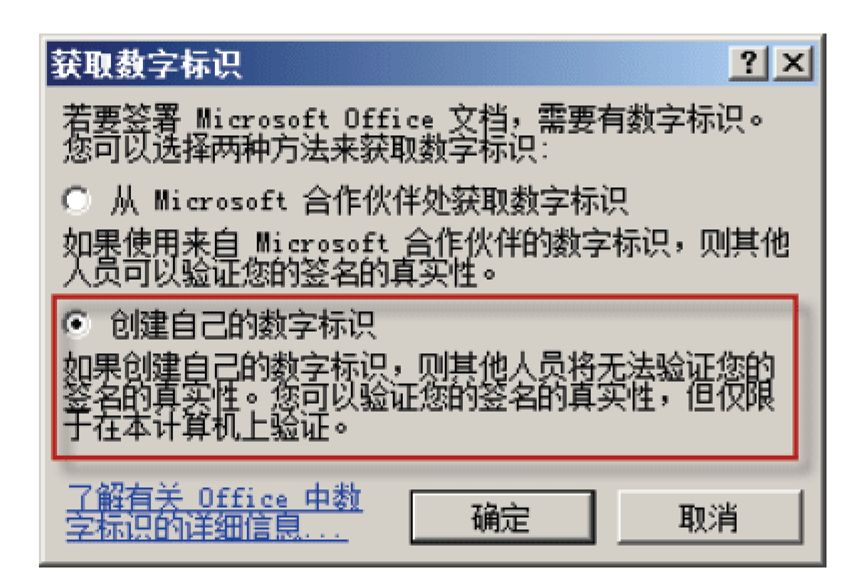


图 2.35 “获取数字标示”对话框的设置

🔔注意：在签署签名时，如果文档没有保存，Excel 会给出提示对话框，提示添加签名前，工作簿需要保存。按照要求将文档保存即可。另外，在第一次设置数字标识来源后，以后再进行数字签名时将直接打开“签名”对话框，而不会再打开“获取数字标识”对话框。

（6）至此，文档的数字签名添加完成。添加数字签名后的文档效果如图 2.38 所示。



图 2.36 “签名”对话框的设置



图 2.37 “签名确认”对话框

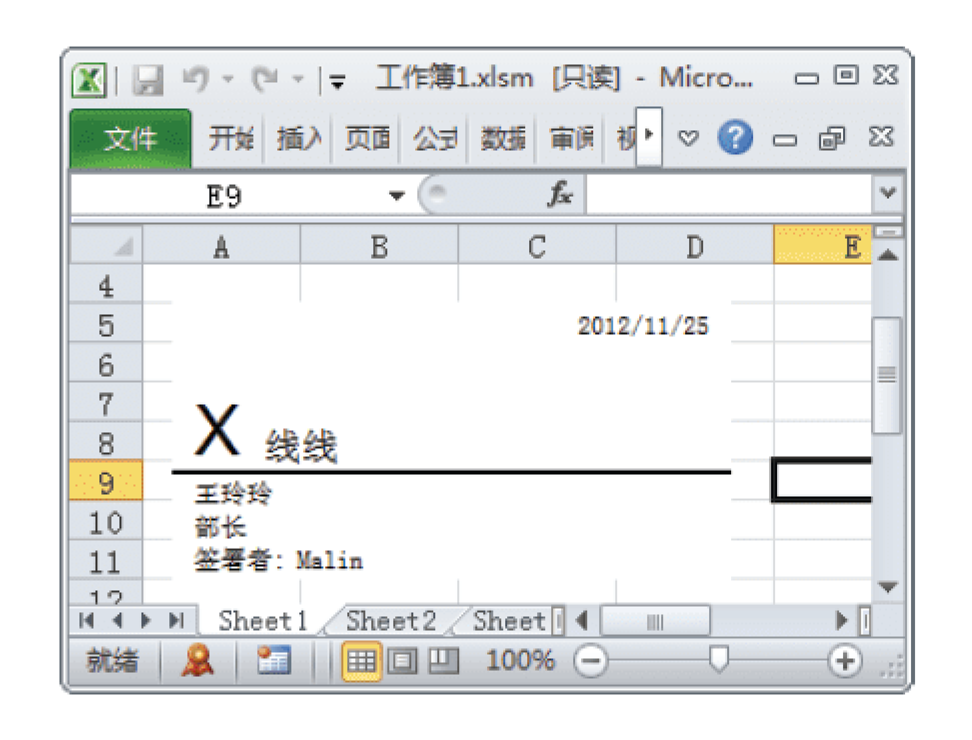


图 2.38 添加数字签名后的效果

🔔提示：Excel 2010 的数字签名支持使用图像和使用手写板的手写墨迹签名。在“签名”对话框中单击“选择图像”按钮可打开“选择签名图像”对话框。使用该对话框选择需要使用的图像后关闭对话框即可为签名添加图像。

2.5.2　使用数字签名

Excel 2010 的宏创建者，可以为宏项目添加数字签名来对宏进行保护。此种签名证书能够确认宏来自于签名者，同时确认宏未被改动。下面介绍使用数字签名来保护宏的方法。

（1）选择“开始→所有程序→Microsoft Office→Microsoft Office 工具→VBA 项目的数字证书”命令，如图 2.39 所示。

（2）在打开的“创建数字证书”对话框中输入证书的名称后单击“确定”按钮，如图 2.40 所示。此时，系统给出新建证书成功提示对话框，如图 2.41 所示。单击“确定”按钮关闭该对话框，完成创建数字证书的工作。

注意：数字签名证书分为商用和个人 2 种类型。商用数字签名需要由独立的证书公司签发，这些公司可以在微软的网站上查到。个人也可以根据需要自行签署，就像本例这样。证书要真正做到来源可靠，还是需要向独立的、公正且可以信赖的第三方组织（即认证中心）申请数字证书。但对于普通用户来说，个人数字签名证书已经能够做到比较安全了。

（3）启动 Excel 2010，打开一个带有宏的文件。按“Alt+F11”键打开 Visual Basic 编辑器。选择“工具”→“数字签名”命令打开“数字签名”对话框。单击对话框中的“选择”按钮打开“选择证书”对话框，在对话框中选择刚才创建的数字证书，如图 2.42 所示。

（4）如果需要查看选择证书，可以进行如下操作。在“选择证书”对话框中单击“查看证书”按钮打开“证书”对话框，选择“详细信息”选项卡，即可在打开的选项卡中查看证书的有关信息。如图 2.43 所示。

（5）完成数字证书的指定后，关闭“选择证书”和“数字签名”对话框，保存当前的 Excel 文档。



图 2.39　选择“VBA 项目的数字证书”命令

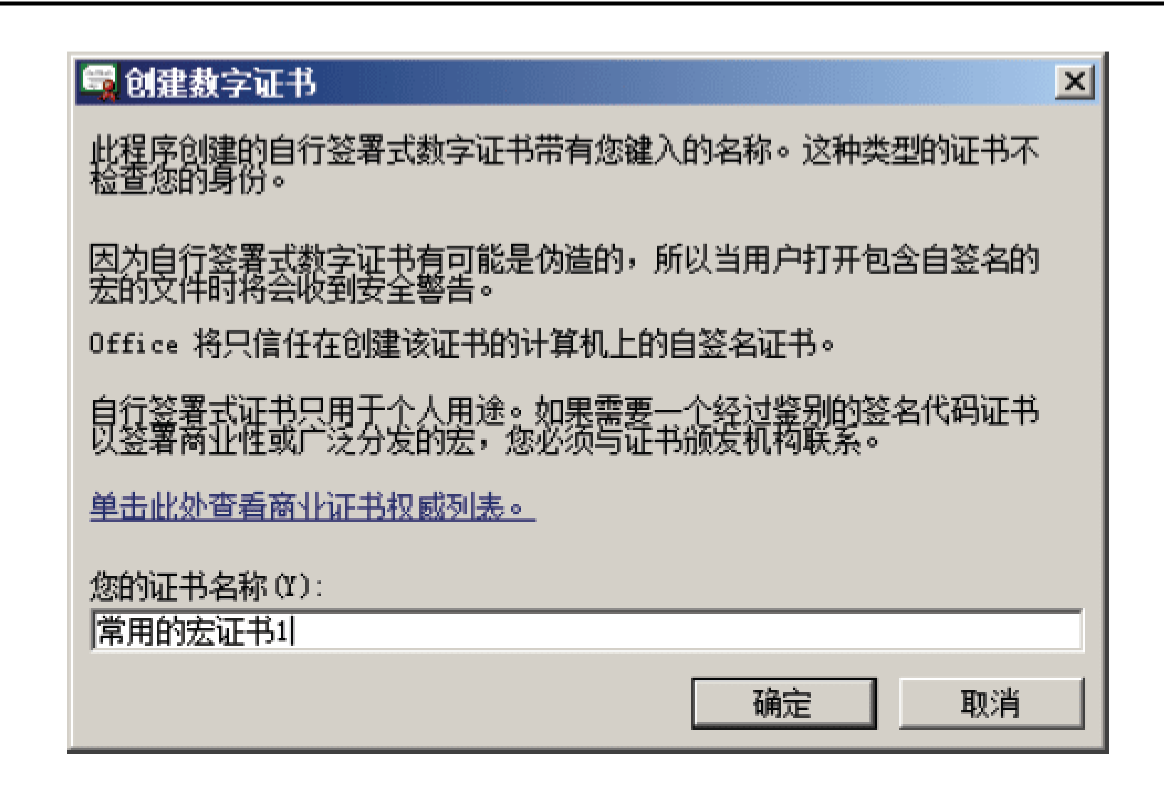


图 2.40　输入证书名称

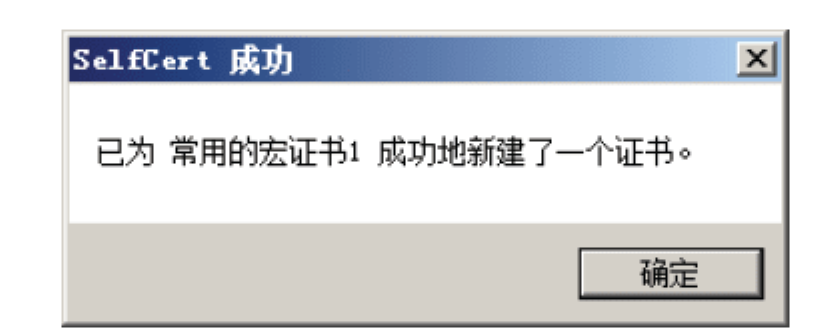
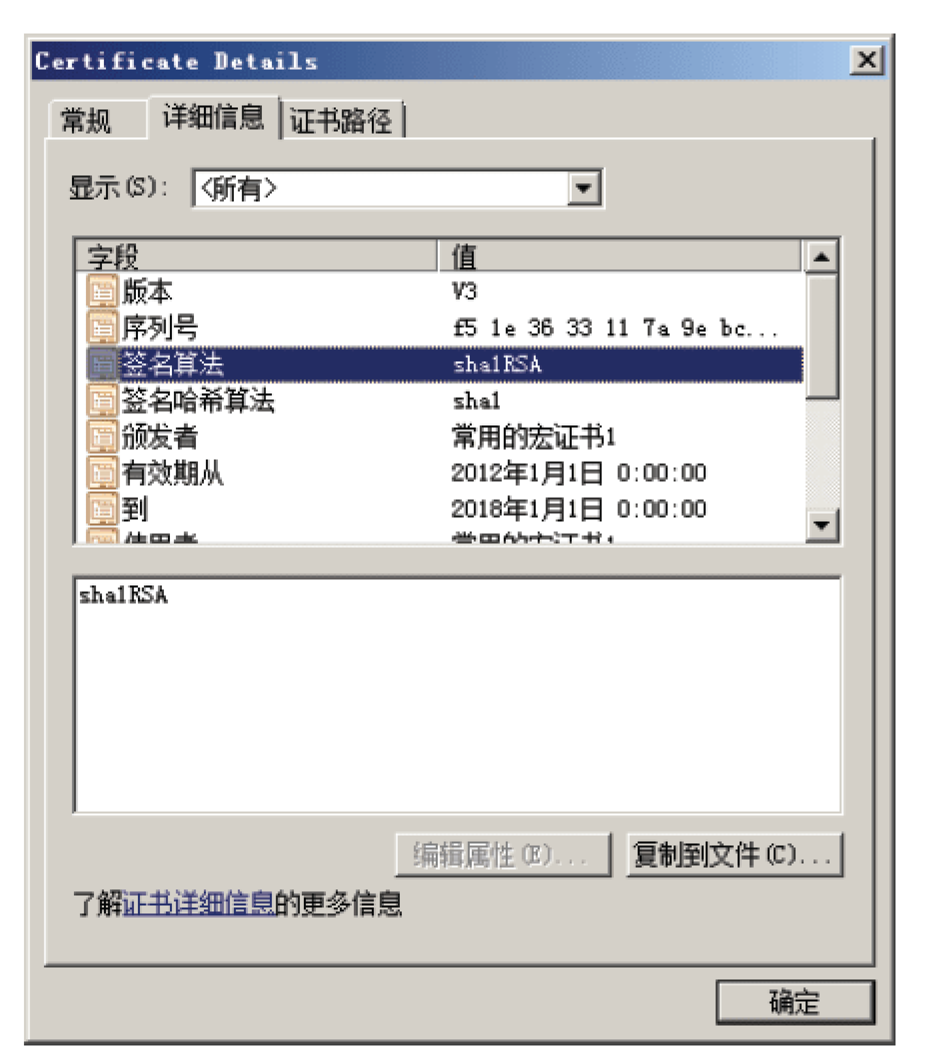


图 2.41　提示新建证书成功



图 2.42　选择数字证书

（6）当第一次打开该文件时，Excel 2010 出现禁用宏的安全警告，选择“文件→选项”命令将会打开“Excel 选项”对话框，选择“信任中心”选项，再单击“信任中心设置”按钮，即可以根据需要选择是否信任此签名。如果信任来自此用户的文档，可以单击“信任对 VBAI 程对象模型的访问”单选按钮选择此项，如图 2.44 所示。单击“确定”按钮完成设置后，以后打开使用此数字签名的文档都将会被认为是可靠的来源，功能区中将不再出现安全警告。



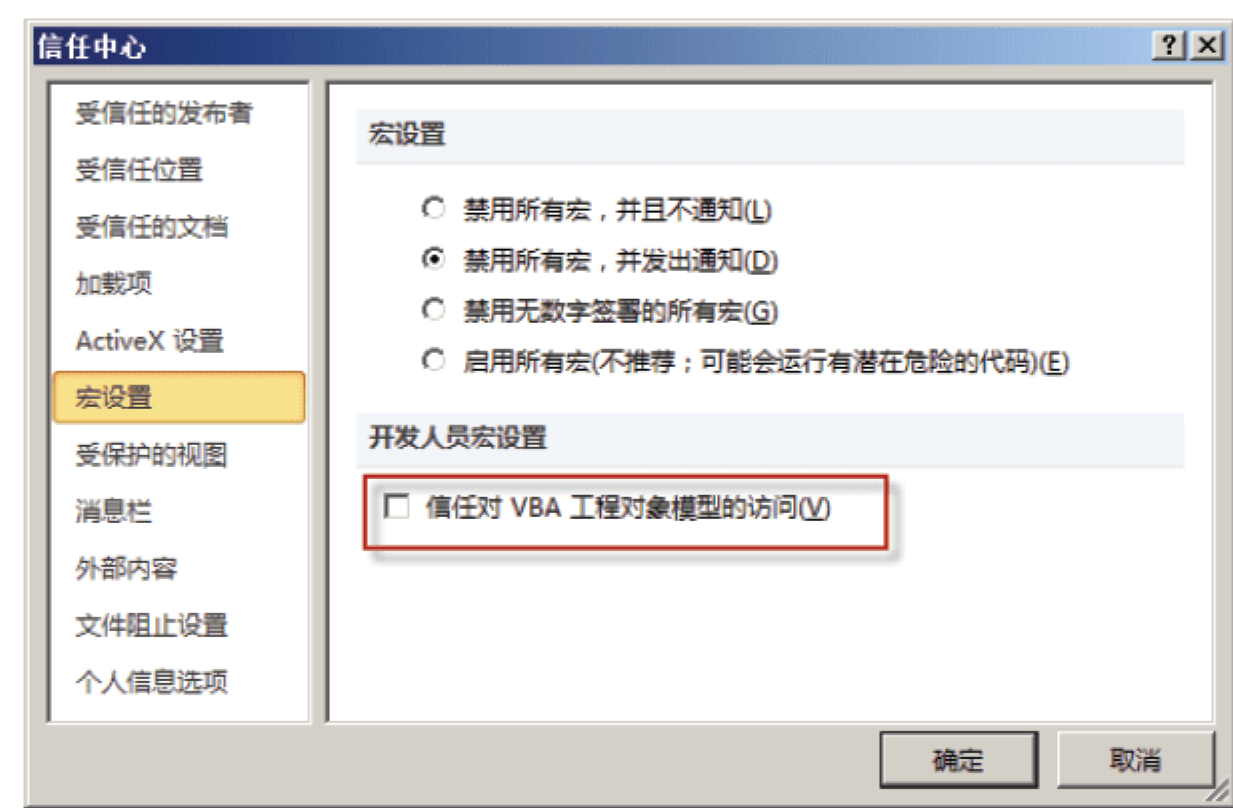


图 2.43　查看和编辑证书的属性　　　图 2.44　“Microsoft Office 安全选项”对话框的设置

提示：对宏添加数字签名后，是否出现宏安全警告要看宏安全设置。只有在“信任中心”对话框中选中“禁用所有宏，并发出通知”或“禁用无数字签署的所有宏”单选按钮时，才会出现宏安全警告。

2.6　小　　结

本章介绍了 Excel 2010 程序设计的基础知识、宏，以及 Excel 安全防护的有关知识。通过本章的学习，读者能够掌握 Excel 2010 中宏的录制、宏的运行以及使用加载宏和宏的安全设置的知识，能够灵活使用宏来减少重复工作，提高工作效率。

对于 Excel 2010 来说，宏实际上是一种无参数的 VBA 子过程，它能帮助用户增强 Excel 的功能，提高数据的处理速度和效率。宏在 Excel 数据表格的创建、数据的处理以及表格功能提升上的意义是明显的。但宏在功能上也有很大的局限性，其无法进行逻辑判断，交互性差，无法创建更复杂的工具栏，无法显示对话框和用户自定义窗体。正是由于这些局限性，使用 VBA 来编程就是进一步提升 Excel 能力的有效手段了。从第 3 章开始，将要正式开始 VBA 的学习，进入 VBA Excel 程序开发的神圣殿堂。

2.7　本 章 习 题

1．在录制新宏时，要将录制的宏保存在新工作簿中，应该执行下面哪个操作？（　　）

A．在“新建宏”对话框中“保存在”下拉列表中选择“个人宏工作簿”选项
B．在“新建宏”对话框中“保存在”下拉列表中选择“新工作簿”选项
C．在“新建宏”对话框中“保存在”下拉列表中选择“当前工作簿”选项
D．直接录制宏，不需要进行任何设置

2．下面哪组快捷键能够作为启动宏的快捷键？（　　）

A．Alt+L　　B．Shift+L　　C．Alt+Shift+L　　D．Ctrl+L

3．使用 Excel 2010 中的“信任中心”对话框设置宏安全性时，应在对话框左侧窗格中选择下面哪个选项？（　）

A．“受信任位置”　B．“加载项”　C．“ActiveX 设置”　D．“宏设置”

4．Excel 加载宏文件的文件后缀名________。

5．启用宏的 Excel 2010 文件后缀名是________。

6．什么是加载宏？

7．如何录制宏？

第 3 章　开发 VBA 的工具

Visual Basic 编辑器是 Excel VBA 的开发环境，该编辑器由工程窗口、属性窗口、代码窗口等多个窗口构成，Visual Basic 编辑器中带有各种代码编辑助手，便于程序编辑，同时提供了很多调试 VBA 程序的调试工具。用户也可以根据自己的使用习惯设置 Visual Basic 编辑器的开发环境，本章的主要学习内容和学习目的有：

- ❑ 了解 Visual Basic 编辑器的界面构成，掌握界面操作的一般方法；
- ❑ 掌握 Visual Basic 编辑器常用窗口的使用，能够初步使用常用窗口解决简单问题；
- ❑ 了解 Visual Basic 编辑器代码输入的特点，能够灵活应用代码输入的各种快捷功能；
- ❑ 了解定制 Visual Basic 编辑器的一般方法，掌握使用断点调试程序的方法。

3.1　Excel 中的 Visual Basic 编辑器

Excel 应用程序的开发离不开 VBA 编程环境，这个编程环境是由 Visual Basic 编辑器（即 VBE）提供的。VBE 是一个独立的应用程序，有自己的独立的操作窗口，能够与 Excel 无缝结合。但是 VBE 环境不能独立打开，必须在启动 Excel 或其他宿主程序后才能打开运行，本节将对 Excel 的 Visual Basic 编辑器进行介绍。

3.1.1　打开编辑器

VBA 实际上是 Visual Basic 的一个子集，其与 Visual Basic 一样是属于面向对象的编程语言，其语法结构与 VB 基本相同，具有与 Visual Basic 基本相同的对象集。VB 所支持的对象属性和方法，VBA 同样支持，只是在事件和属性的特定名称上略有差异。VBA 集成诞生于 VB 之后，其继承了 VB 很多的对象、属性和方法。由于 VBA 是 Office 办公软件内嵌的编程语言，也有人将其称为“寄生在 Office 中的 Visual Basic”。

在 Excel 2010 中，编写 VBA 代码、调试宏以及应用程序开发等操作都离不开 Visual Basic 编辑器，使用 Visual Basic 编辑可以完成创建 VBA 过程、创建 VBA 用户窗体、查看或修改对象属性以及调试 VBA 程序等任务。在 Excel 中启动 VBA 编辑器一般有下面两种方法：① 在 Excel 2010 中，可以单击“开发工具”选项卡中“Visual Basic”按钮来打开 Visual Basic 编辑器，如图 3.1 所示。② 在 Excel 2010 工作表标签上右击，在弹出的快捷菜单中选择“查看代码”命令，同样能打开 Visual Basic 编辑器，如图 3.2 所示。

提示：Excel 2010 中打开 Visual Basic 编辑器的方法很多，按 Alt+F11 键能够快速打开 Visual Basic 编辑器。如果在工作表中创建了宏，在“开发工具”选项卡中单击“宏”按钮打开“宏”对话框，单击对话框中的“编辑”或“单步运行”按钮均能够打开 Visual Basic 编辑器。

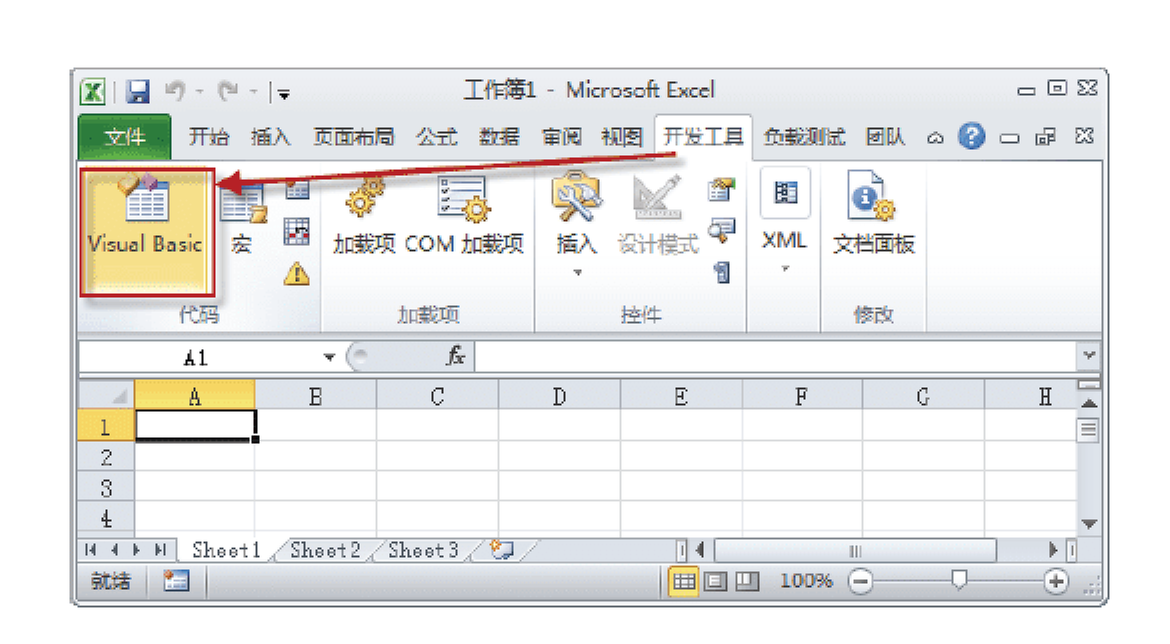


图 3.1　单击 Visual Basic 按钮启动 Visual Basic 编辑器

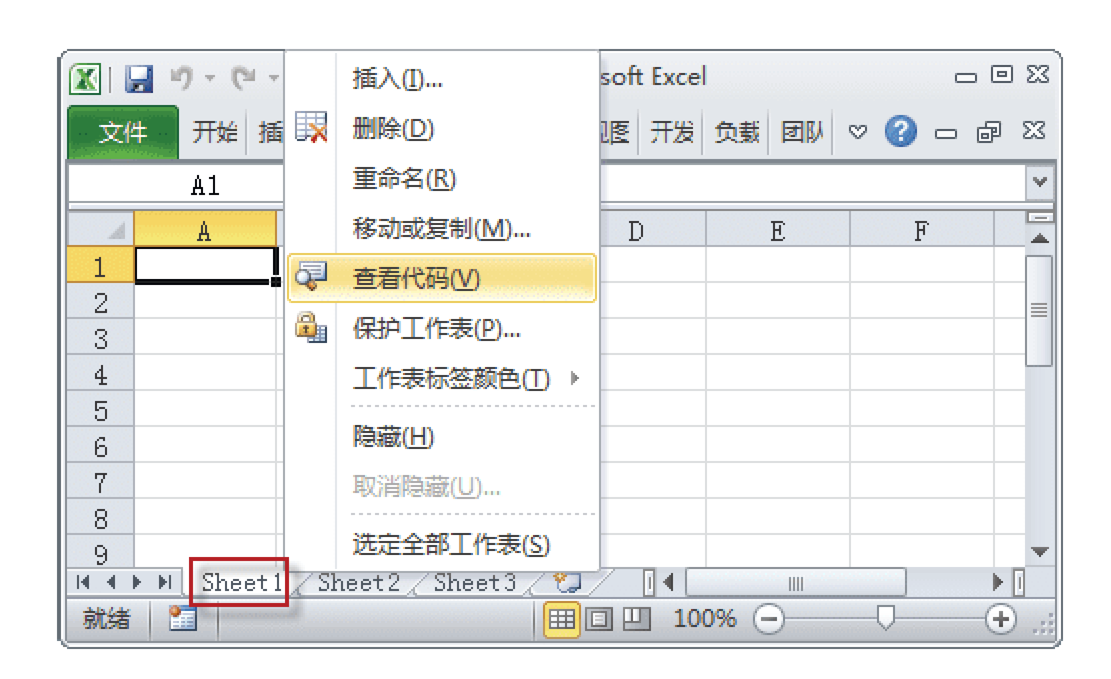


图 3.2　选择快捷菜单中的“查看代码”命令

3.1.2　剖析编辑器

启动 Visual Basic 编辑器后，可以看到 Visual Basic 编辑器的界面构成，如图 3.3 所示。编辑器的界面可以根据任务的不同进行定制，也可以隐藏一些子窗口或更改子窗口的位置和大小。

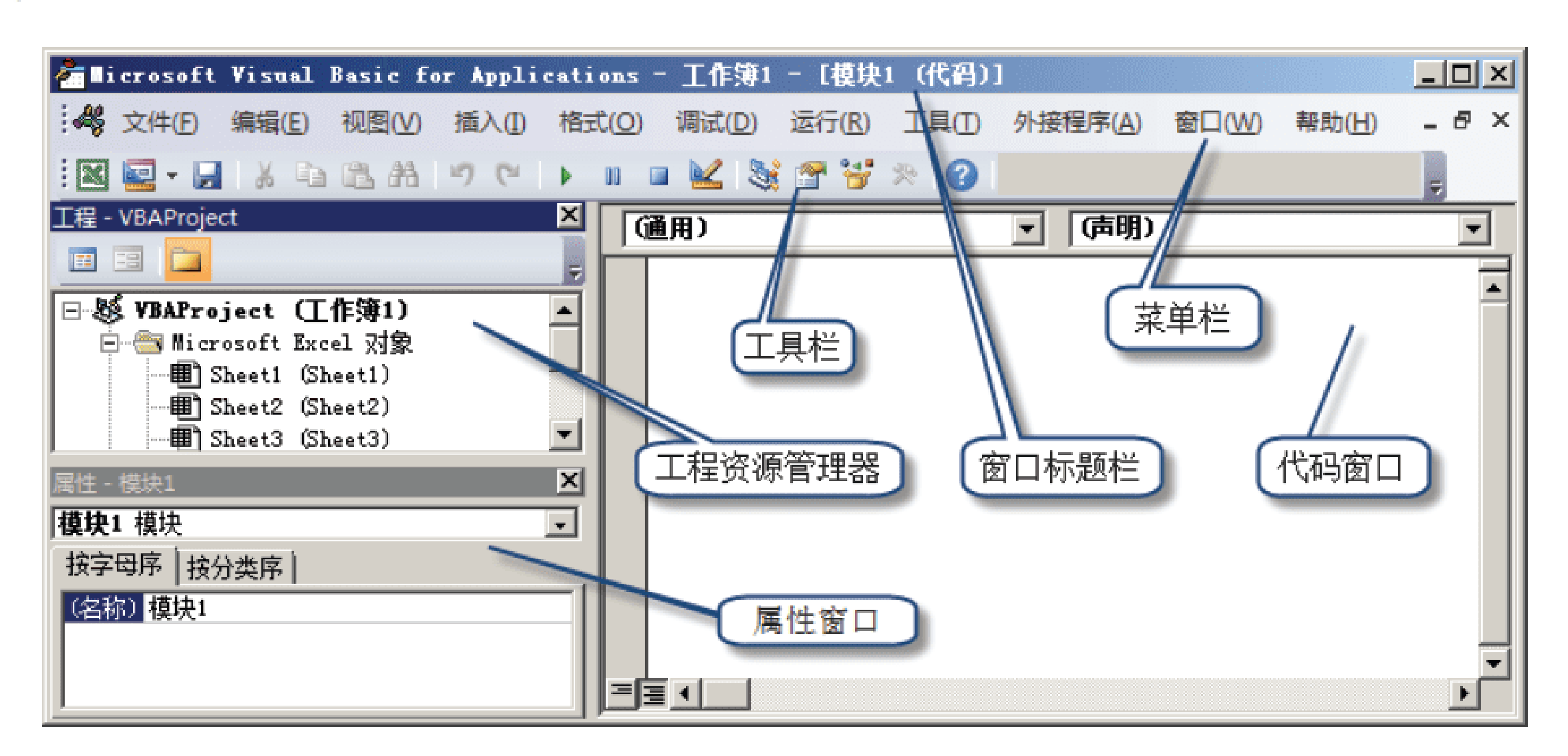


图 3.3　Visual Basic 编辑器窗口

Excel 2010 的 Visual Basic 编辑器仍然采用以往版本中常用的菜单启动方式，主界面采用与其他 Windows 应用程序相同的菜单栏和工具栏结构。菜单栏提供了用于操作的绝大多数菜单命令，通过选择菜单中的菜单命令来执行相应的操作。

Visual Basic 编辑器共有 6 个工具栏，在默认情况下标准工具栏位于菜单栏的下方，其他工具栏处于隐藏状态。如果需要使用隐藏的工具栏，可以通过选择“视图”→“工具栏”→“自定义”命令打开“自定义”对话框，如图 3.4 所示。在对话框的“工具栏”选项卡中选择需要使用的工具栏选项，选择的工具栏即可显示。

🔔**提示：** Visual Basic 编辑器中的工具栏与 Excel 环境中的工具栏一样，也可以实现移动、停靠在界面的边界、显示和隐藏等操作。

Visual Basic 编辑器中包含有各种子窗口，这些子窗口可以通过拖动改变其在 Visual Basic 编辑器窗口中的位置，可以通过拖动边框改变其大小。选择“视图”菜单中的菜单命令可以显示隐藏的子窗口，同时单击子窗口标题栏上的关闭按钮可关闭此子窗口。拖动这些子窗口到主界面的边界处，能够使其停靠在主界面的边界。如：拖动“立即窗口”，将其停靠在主界面的右侧，如图 3.5 所示。

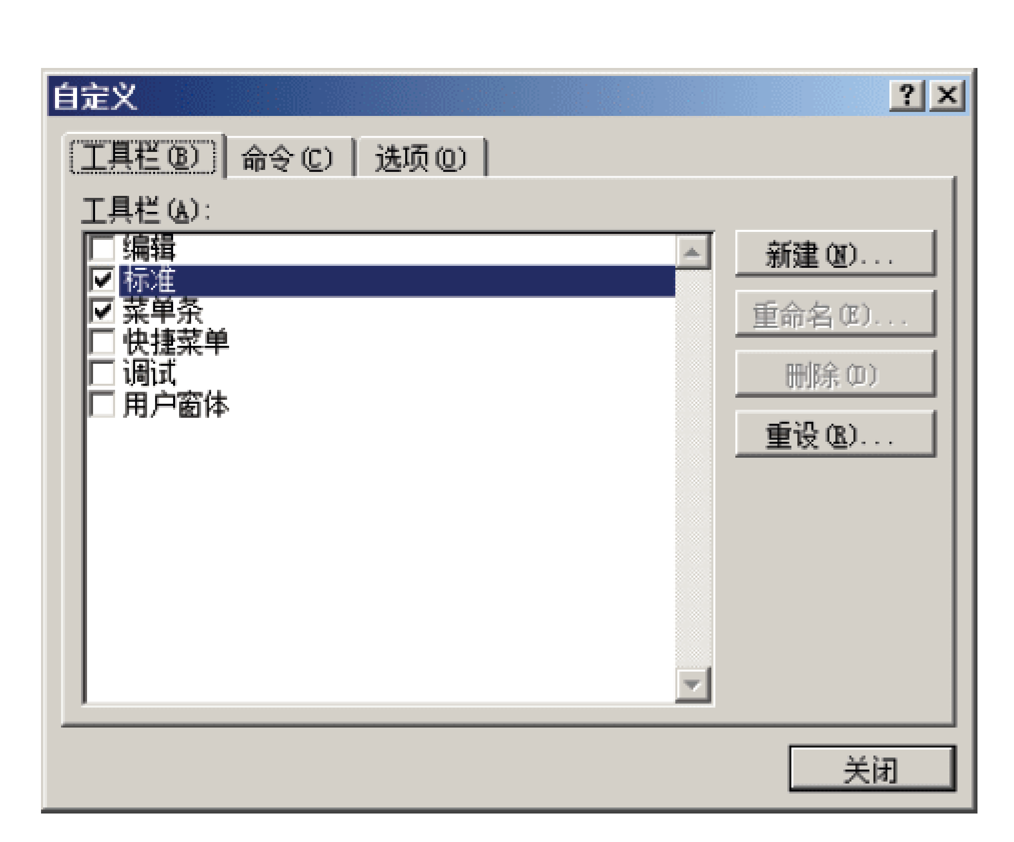



图 3.4　“自定义”对话框

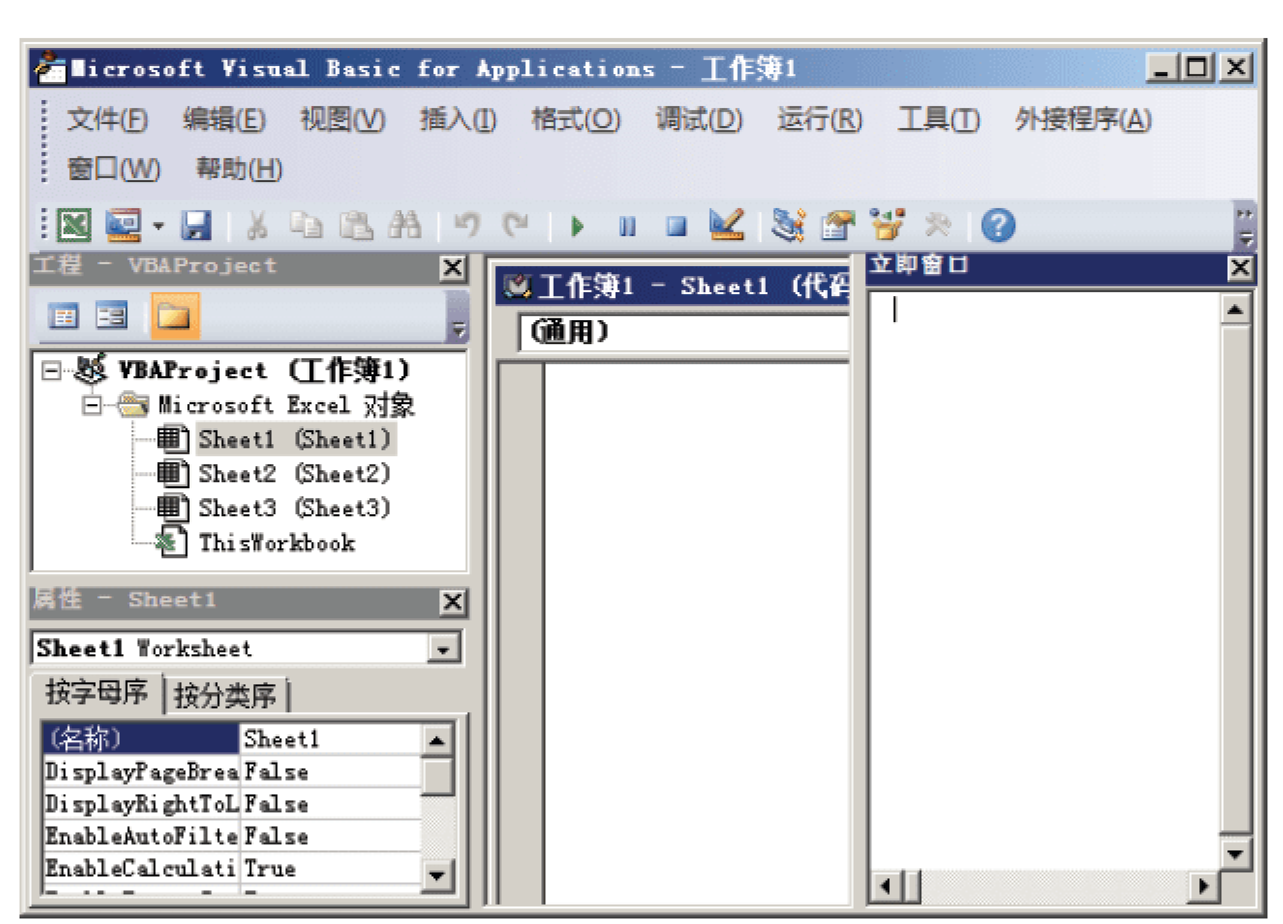


图 3.5　“立即窗口”停靠在主界面的右侧

3.2　常用编辑器窗口

在 Visual Basic 编辑器中存在着大量的子窗口，用户在这些子窗口中可以完成代码的编写、对象属性的修改以及程序的调试等工作。在本节将分别对 Visual Basic 界面常用的子窗口的功能及其使用进行详细的介绍。

3.2.1　使用工程窗口查看工程结构

在 Excel 中，每一个工作簿就是一个工程，工程的默认名称为 VBA Project（工作簿名称）。一个工程一般包括 4 类对象，即：Excel 对象、窗体对象、模块对象和类对象。其中，Excel 2010 对象包括一组 Sheet 对象和 ThisWorkbook 对象，每一类对象都可以包括多个具体的对象。

为实现对工程的管理，Visual Basic 编辑器提供了一个工程资源管理器，即停靠于主界面左侧的“工程”窗口。在“工程”窗口中，以树形结构来管理各个工程。单击工程名左侧的⊞将工程展开，而单击展开工程左侧的⊟可以将展开的工程重新折叠，如图 3.6 所示。

“工程”窗口上方包含 3 个按钮，单击左侧的“查看代码”按钮，可以打开当前选择模块的“代码”窗口。单击中间的“查看对象”按钮，可以显示 Excel 对象文件夹中

所选择的工作表或窗体文件夹中的窗体。单击右侧的“切换文件夹”按钮，可以隐藏或显示工程窗口中的文件夹。

使用“工程”窗口能够对工程中的对象进行管理，管理包括对象的插入和删除、模块的导入和导出等。下面以对模块的操作为例来介绍具体的操作方法。

（1）插入模块。在“工程”窗口的任意一个对象上右击，在弹出的快捷菜单中选择“插入”→“模块”命令，即可在当前工程中插入一个模块，如图 3.7 所示。

图 3.6 “工程”窗口

（2）移除模块。工程中的模块如果需要删除，可以在要删除的模块上右击，然后单击弹出式菜单中的“移除模块”命令即可，如图 3.8 所示。

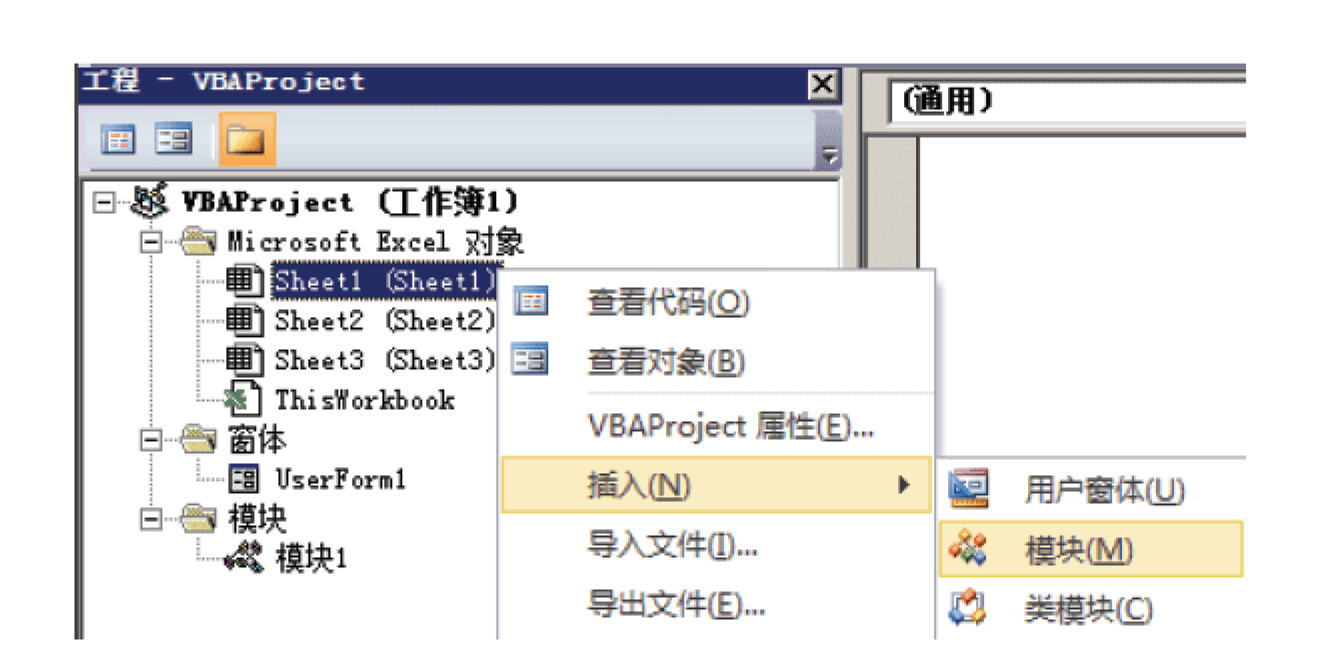


图 3.7　插入模块

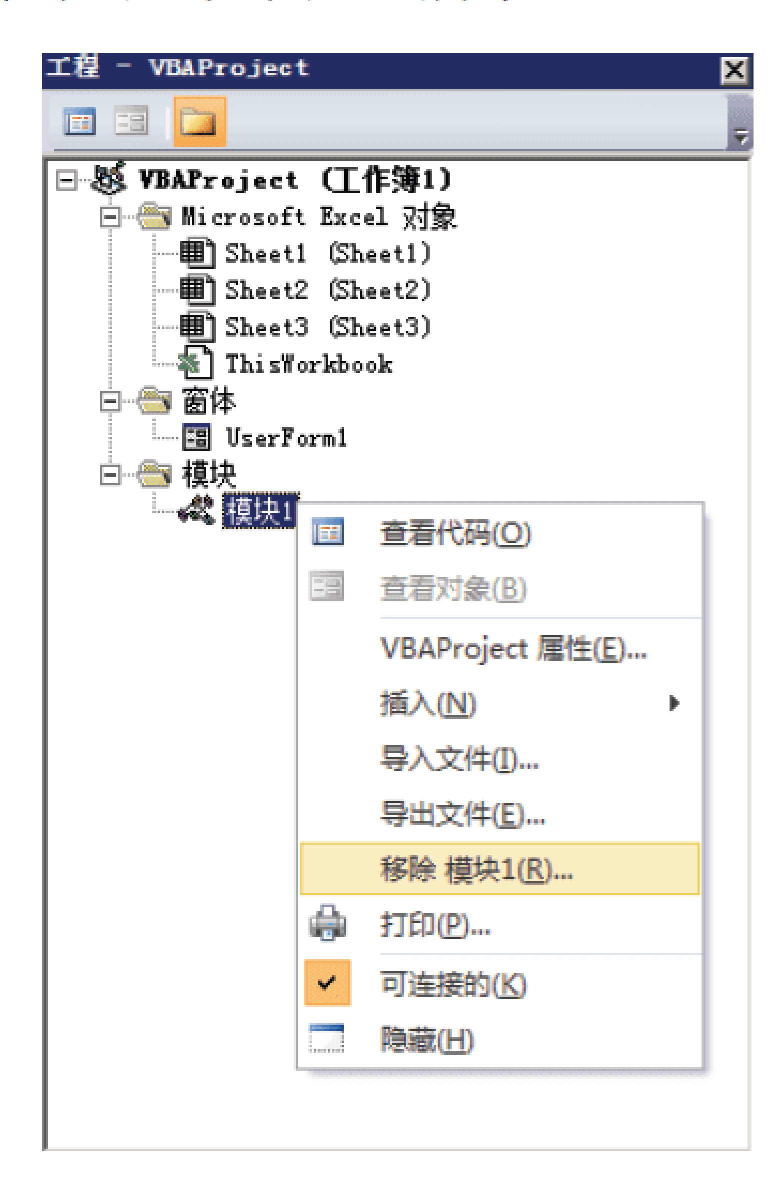


图 3.8　移除模块

（3）导出模块。如果开发的模块需要保存，可以将模块导出。方法是在需要导出的模块上右击，在弹出的快捷菜单中选择“导出文件”命令。此时可以打开“导出文件”对话框。在对话框中选择导出文件的保存位置并输入导出文件的文件名，单击“保存”按钮即可将模块保存，如图 3.9 所示。

提示：“导出文件”命令不仅能够用于模块的保存，也可用于其他对象的保存。对于模块文件来说，其扩展名为“.bas”。对于窗体文件来说，其扩展名为“.frm”。对于类来说，其扩展名为“.cls”。这里要注意，导出的模块文件实际上是一个文本文件，可以用记事本打开并查看其内容。

（4）导入模块。在“工程”窗口中右击，在弹出的快捷菜单中选择“导入文件”命令可以打开“导入文件”对话框。在该对话框中选择需要导入的文件，单击“打开”按钮即可将文件导入到“工程”窗口中，如图 3.10 所示。

提示：“工程”窗口中显示的对象支持拖动操作。如将模块从一个工程拖动到另一个工程中，能够实现该模块的复制。



图 3.9　“导出文件”对话框的设置　　　　图 3.10　“导入文件”对话框

3.2.2　使用属性窗口查看工程属性

“属性”窗口用于设置对象的属性。面向对象的程序设计一个重要步骤就是对对象的属性进行设置。设置对象的属性一般有两种方法，一种是通过编程，使用程序代码来更改对象的属性。另一种方法是在“属性”窗口中直接更改对象的属性。下面以更改用户窗体的属性为例来介绍设置对象属性的方法。

（1）在“工程”窗口中插入一个用户窗体，同时选择这个窗体对象，如图 3.11 所示。

（2）首先更改窗体的“Caption”属性，该属性将改变窗体标题栏显示的内容。在“属性”窗口中双击 Caption 选项，在文本框中输入标题文字，如图 3.12 所示。完成输入后按 Enter 键确认输入。

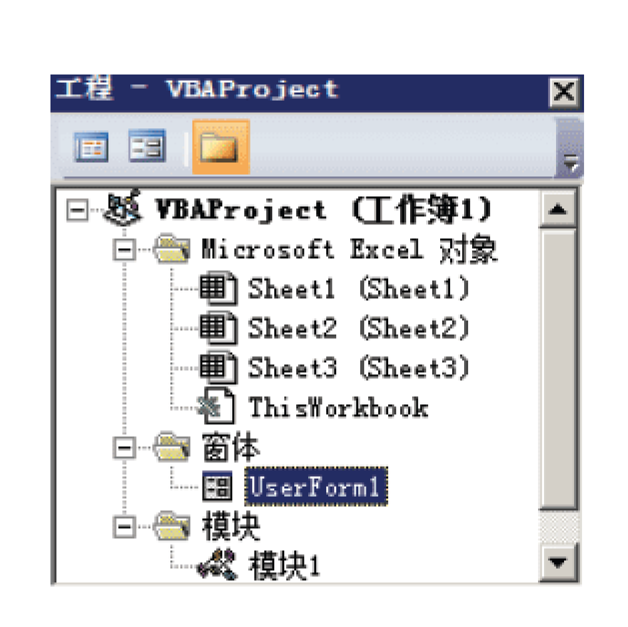


图 3.11　选择创建的用户窗体对象

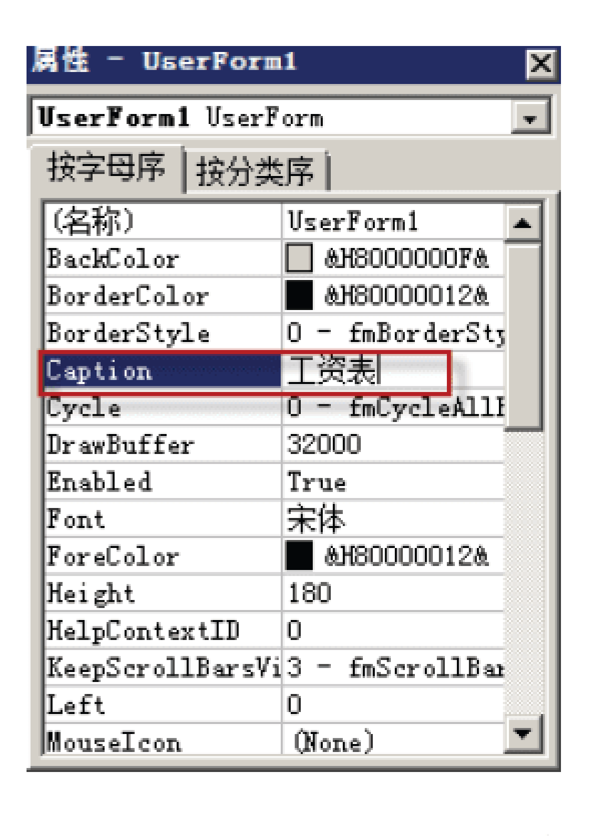


图 3.12　更改 Caption 属性

（3）对象的属性是文件名时，需要打开对话框选择相应的文件。例如，以图片作为用户窗体的背景，则用户窗体的 Picture 属性应该是该背景图片的文件名。此时，单击 Picture 项，然后单击其右侧出现的...按钮即可打开“加载图片”对话框。在对话框中选择作为背景的图片，单击“确定”按钮关闭对话框，如图 3.13 所示。

提示： 在“属性”窗口中，如果需要删除作为背景的图片，可在 Picture 属性栏中单击鼠标，然后按 Delete 键。此时，其属性值由“（Bitmap）”变为“（None）”，背景图片将从用户窗体中删除。



图 3.13　设置 Picture 属性

（4）对象的很多属性都有固定的设置范围，这样的属性值可以通过下拉列表框来进行选择。例如，为窗体添加背景图片后，希望背景图片铺满整个窗体，此时可以通过设置窗体的“PictureSizeMode”属性来实现。具体的操作方法是，单击 PictureSizeMode 属性项，再单击出现的下三角按钮，打开下拉列表框。在下拉列表中选择相应的设置项，完成对属性的设置，如图 3.14 所示。完成上述属性设置后的用户窗体的外观如图 3.15 所示。

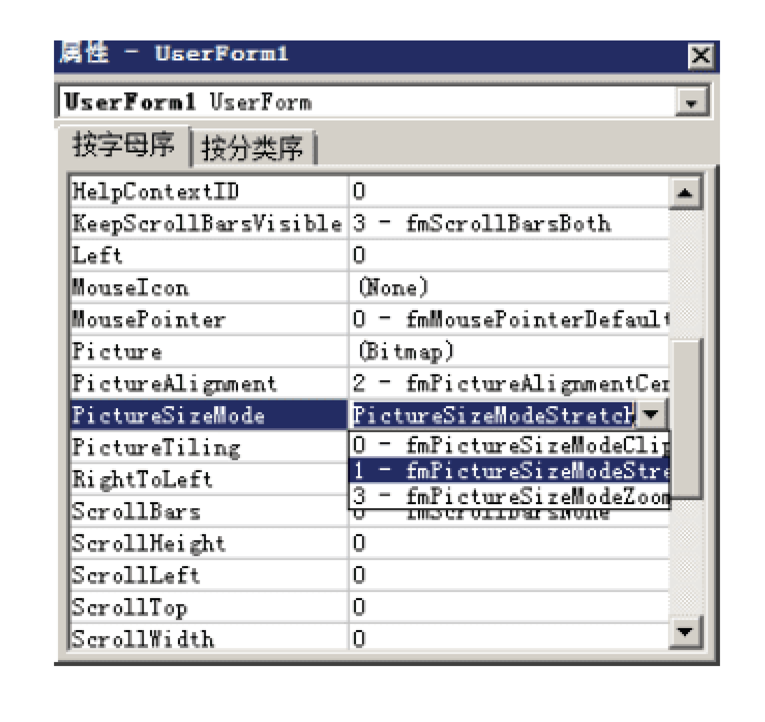


图 3.14　设置 PictureSizeMode 属性

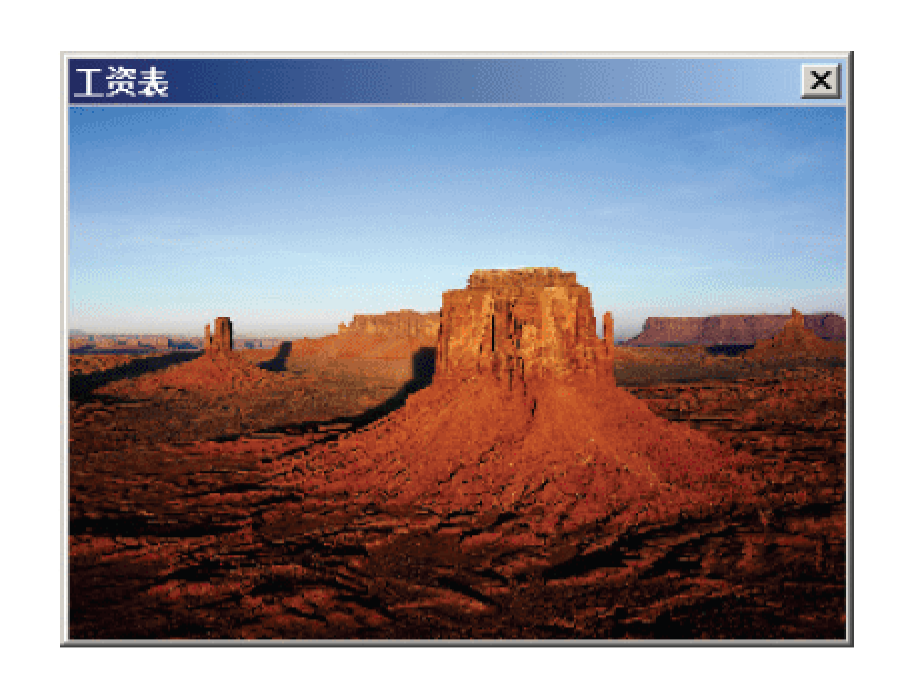


图 3.15　完成属性设置后的窗体效果

3.2.3　使用代码窗口编辑调试 VBA 代码

“代码”窗口用来显示和编辑程序代码。Excel VBA 是以过程的方式来组织程序的，一个过程就是一个完成特点任务的代码的集合。每个工程中的每一个对象都可以编写相关的过程代码，而每一个对象的过程代码的编写都是在自己的“代码”窗口中完成的。在“代码”窗口中编写程序，一般采用以下步骤。

（1）打开“代码”窗口。每一个对象都有自己的“代码”窗口，在“工程”窗口中双击该对象能够打开对象的“代码”窗口，如图 3.16 所示。

（2）添加事件。“代码”窗口的顶部有两个下拉列表框，左侧的“对象”下拉列表用于在当前模块的对象间进行切换，可以用于查看不同对象的事件代码。右侧为“过程/事件”下拉列表框，可以快速选择对象的过程或事件，如图 3.17 所示。

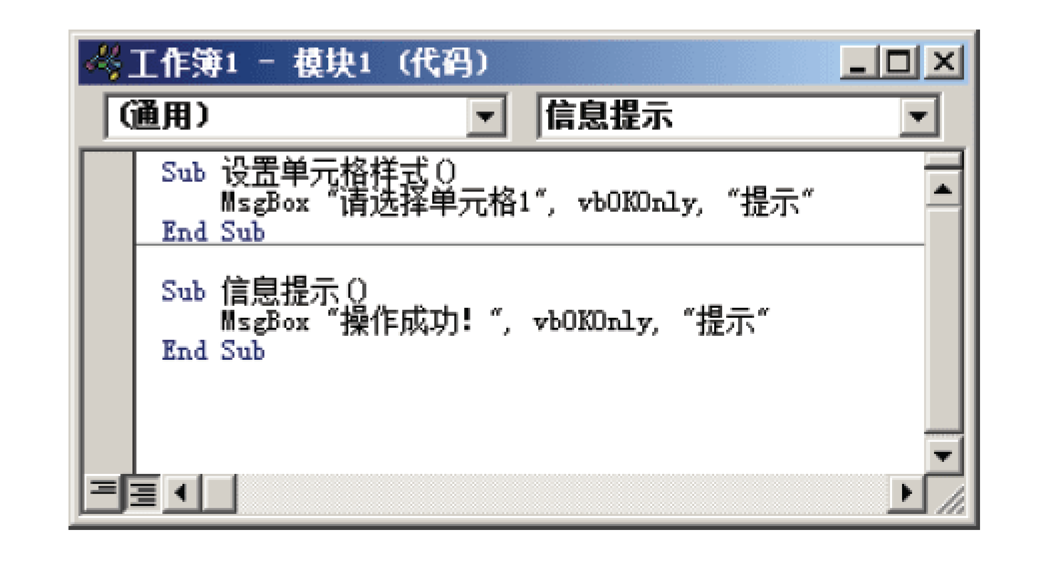


图 3.16 “代码”窗口

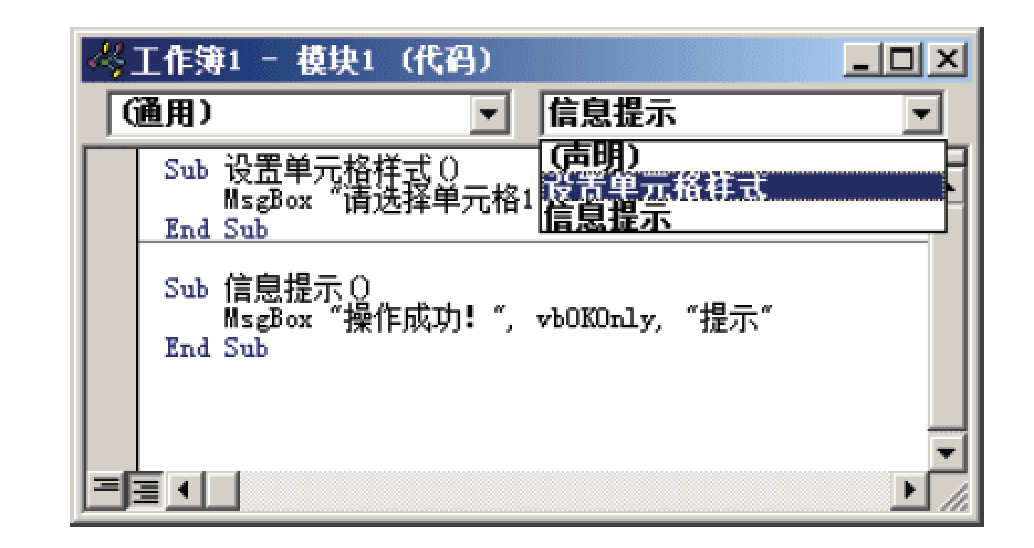


图 3.17　顶部的“对象”和“过程/事件”下拉列表

（3）输入代码时切换视图模式。在“代码”窗口的下方有两个视图按钮，它们分别是“全模块视图”按钮和“过程视图”按钮。当单击“过程视图”按钮时，“代码”窗口将只显示一个过程，如图 3.18 所示。过程的选择可以通过“过程/事件”下拉列表框来进行。当“全模式视图”按钮被按下后，“代码”窗口中将显示模块的所有过程。

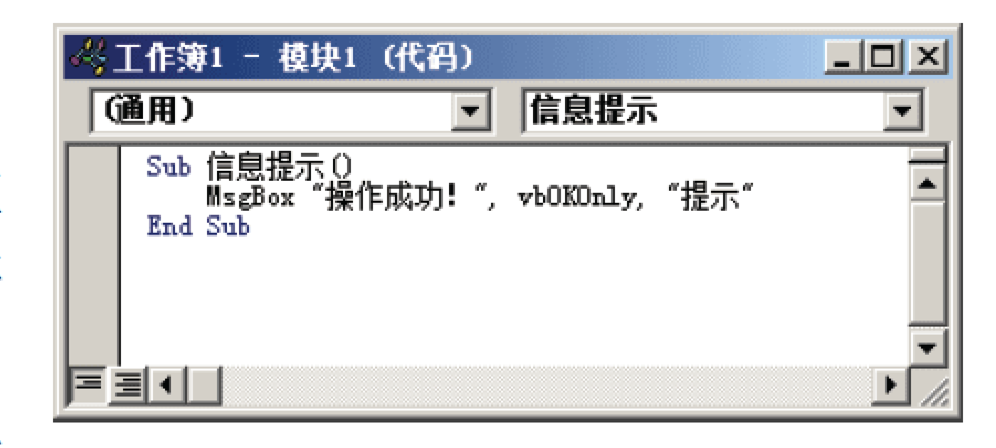


图 3.18 “过程视图”模式

（4）调试程序时设置断点。“代码”窗口的左侧有一个边界标识条，其用来放置中断标志和程序运行标识。在边界标识条上单击，可以放置一个中断标识，如图 3.19 所示。如果需要删除这个中断标识，只需要在中断标识上再次单击即可。在逐语句调试程序时或程序出错时，边界标识条上会出现黄色的箭头以标识当前的语句，如图 3.20 所示。



图 3.19　放置中断标识

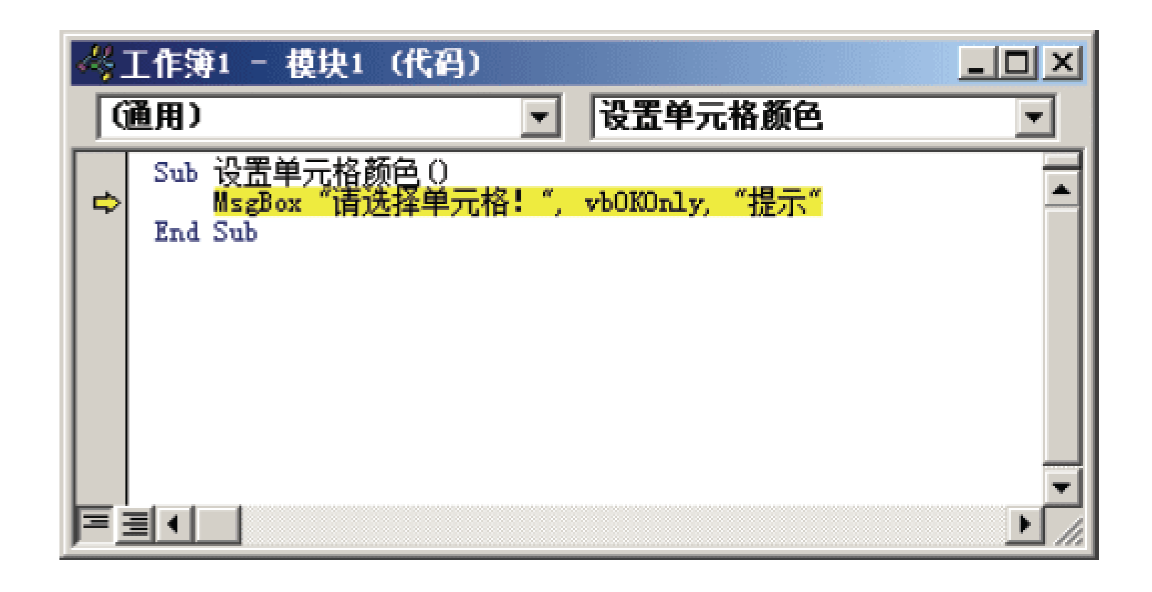


图 3.20　标识程序运行的位置

提示：在编写代码时，过程与过程间会出现一条线，这条线称为过程分隔线。在添加过程后，Visual Basic 编辑器会自动添加过程分隔线。

3.2.4　使用立即窗口查看工程结果

“立即窗口”可用来显示程序运行结果，在默认情况下该窗口是隐藏的，选择“视图”→“立即窗口”命令可以打开“立即窗口”。“立即窗口”的作用主要体现在程序的调试上，使用“立即窗口”可以显示调试语句的信息。如，使用 Degbug.Print 语句可以将变量、表达式或属性语句在“立即窗口”中显示处理。下面通过在“立即窗口”中显示一段代码的运行为例来介绍使用“立即窗口”调试程序的两种方法。

（1）直接输入命令。在“立即窗口”窗口中可以直接输入命令，按 Enter 键后可以直接显示结果。如在“立即窗口”中输入如下代码，按 Enter 键后，当前工作表中的 A1～A3 单元格被选择，效果如图 3.21 所示。

```
Range("A1:A3").Select
```

（2）“代码”窗口中为过程代码添加 Debug 语句，添加的程序代码所示。程序运行时，会在“立即窗口”窗口中显示文字“单元格已经选择！”，如图 3.22 所示。

```
Debug.Print "单元格已经选择！"
```

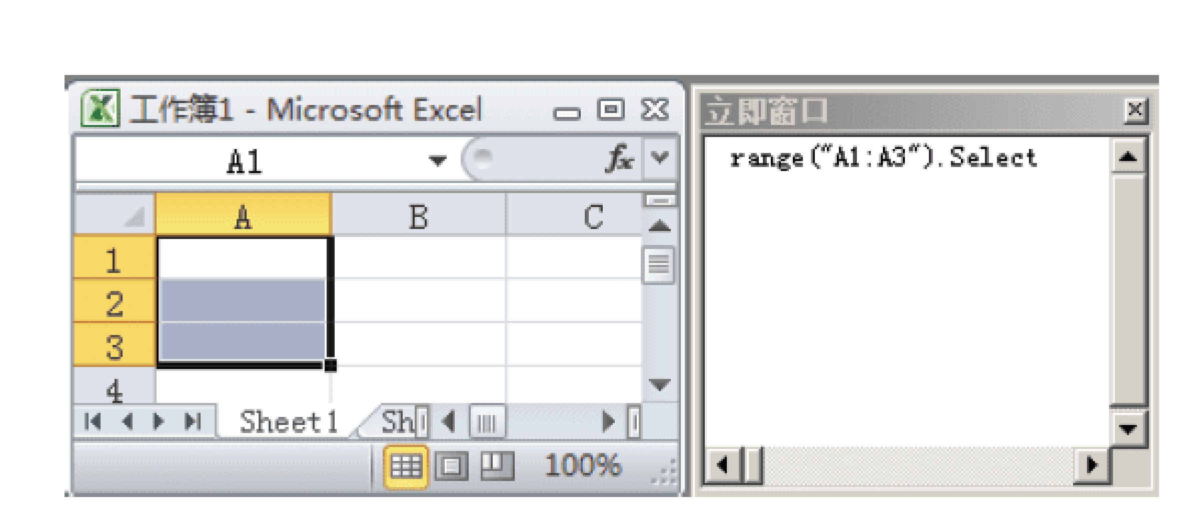


图 3.21　在“立即窗口”窗口中输入代码后的运行效果

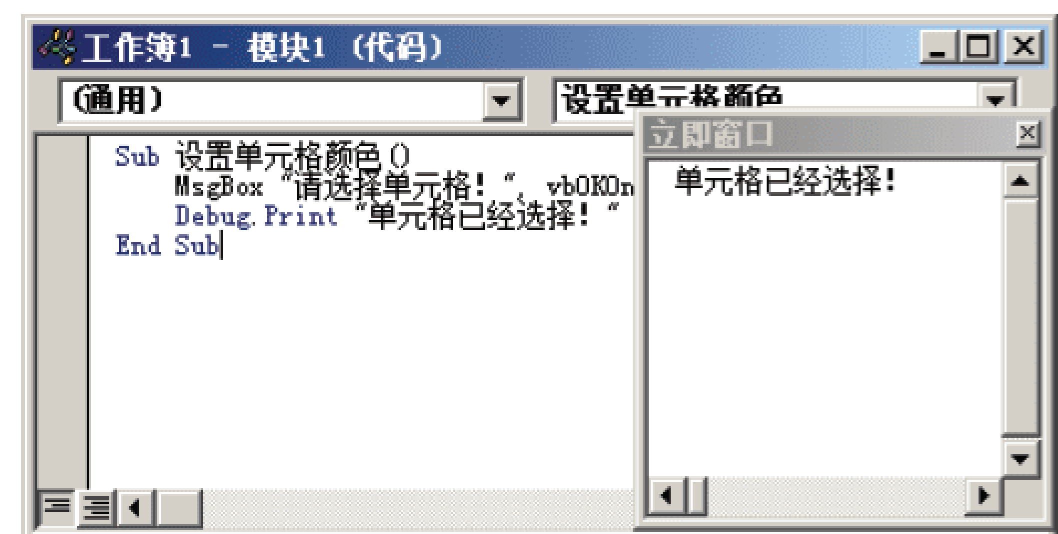


图 3.22　在“立即窗口”窗口中显示文字

提示：在程序编写过程中，“立即窗口”主要用于程序的调试。使用“立即窗口”可以在程序执行时查询或改变变量值和属性值，在代码中调用某个过程，对可能存在问题的代码进行检测。

3.2.5　使用对象浏览器窗口查看所有对象

对于 VBA 初学者来说，掌握数目众多的对象并熟悉对象的属性和方法，不是一件太容易的事情。Visual Basic 编辑器提供了一个对象浏览器，其能够列出 Excel 中的所有对象，显示对象库以及工程过程中可以使用的类、属性、方法、事件以及常量和变量。用户可以使用对象浏览器查询既有的对象，学习对象的使用方法。下面介绍对象浏览器的使用方法。

（1）选择“视图”→“对象浏览器”命令可以打开“对象浏览器”窗口。对象浏览器中包括了 Excel、MSForms、Office、stdole、VBA、VBAProject 共 6 个对象库。在“对象浏览器”窗口的“工程/库”下拉列表框中可以选择需要查询的对象库的类型，如图 3.23 所示。

（2）选择对象库类型后，在“类”列表框中选择需要查询的对象，在右侧的“'AddIn' 的成员”列表框中选择需要查询的成员。此时，在“对象浏览器”窗口下方的“详细数据框”列表框中将显示选择成员的定义，如图 3.24 所示。

提示：在详细数据列表框中，用代码编写的方法、属性、事件或常数将以粗体显示。同时，列表框中还包含一个超链接，可以跳转到该元素所属的类和库。某些成员的跳转可以跳到其上层类。

下面介绍搜索需要的对象或成员的方法。这里搜索的对象是 Excel 库中的 Selection 对

象，需要查看该对象的使用方法。在“搜索文字”下拉列表框中输入需要查询的文字，单击“搜索”按钮，在“对象浏览器”窗口中将出现“搜索结果”列表框。在“搜索结果”列表框中选择需要查询的对象，则可以查看该对象的详细信息，如图 3.25 所示。

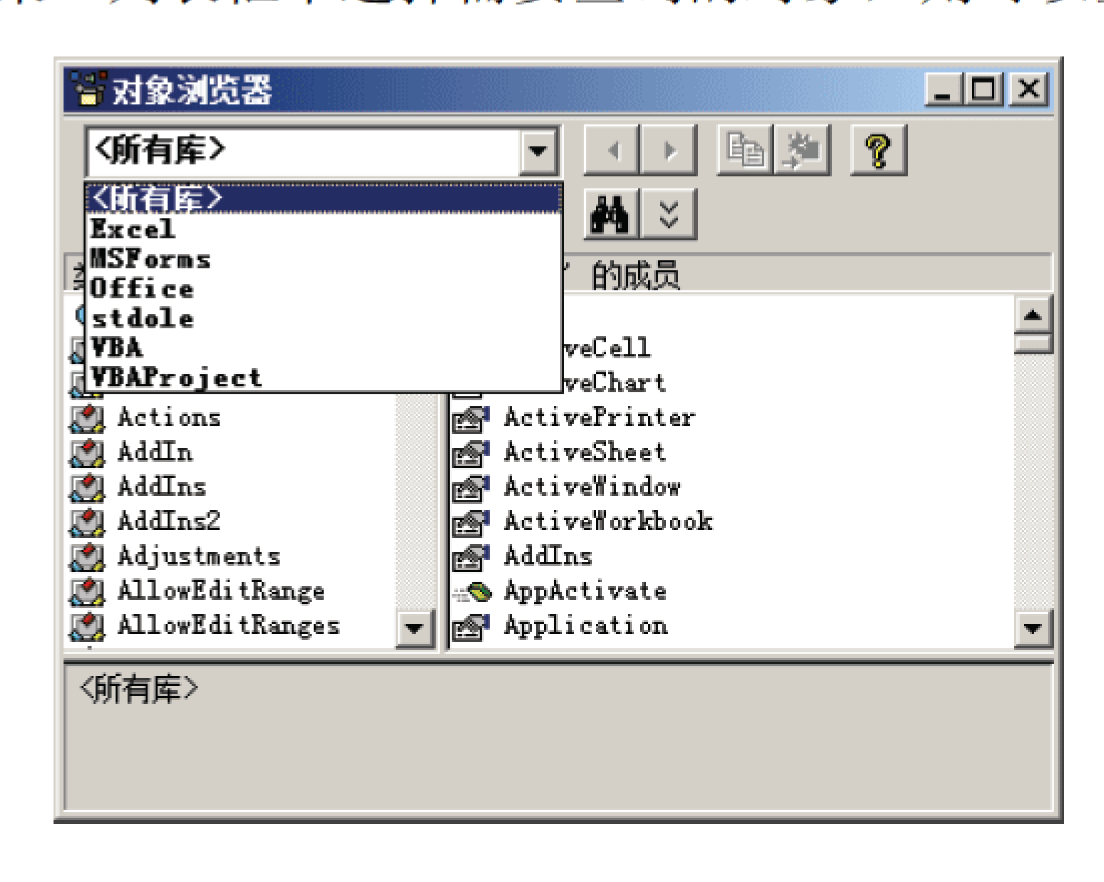


图 3.23　选择对象库

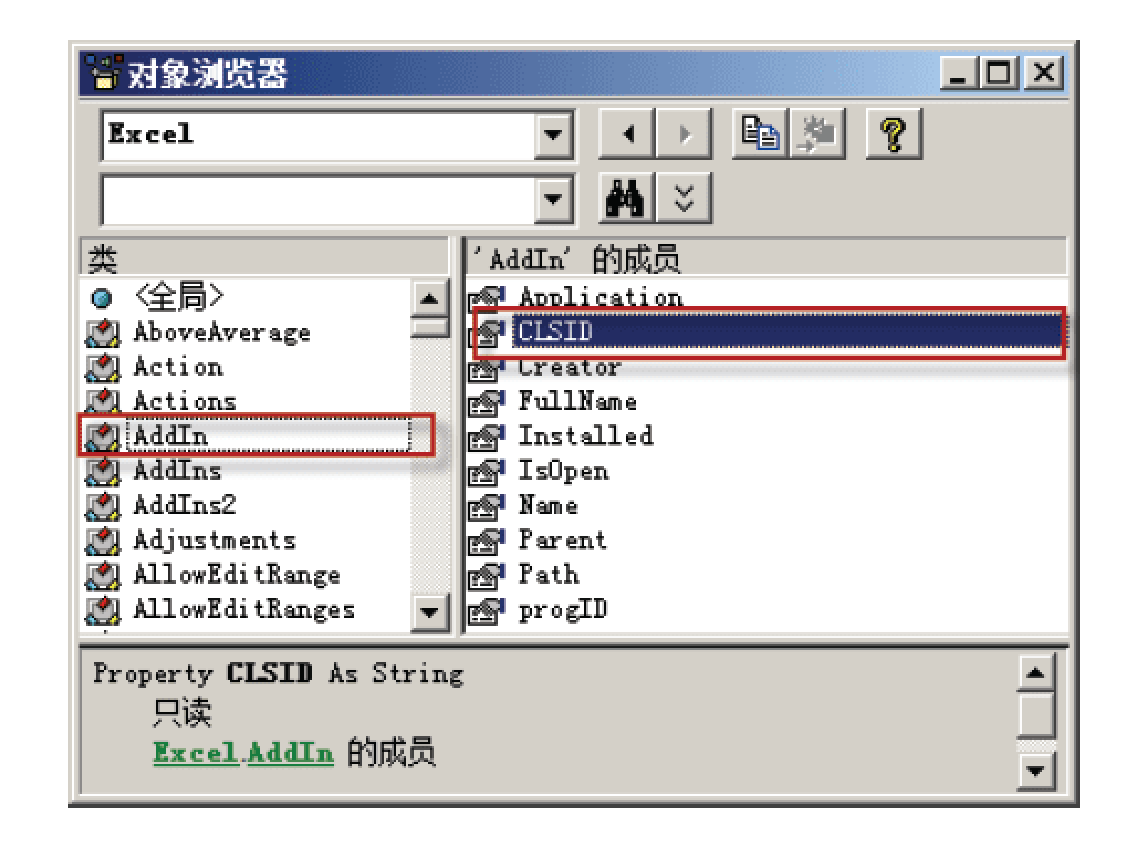


图 3.24　“对象浏览器”窗口

如果在“对象浏览器”窗口中查询到的结果无法解决问题，可以单击窗口工具栏上的“帮助”按钮打开“帮助”窗口，如图 3.26 所示。在“帮助”窗口中将可以得到更为详细的帮助信息。

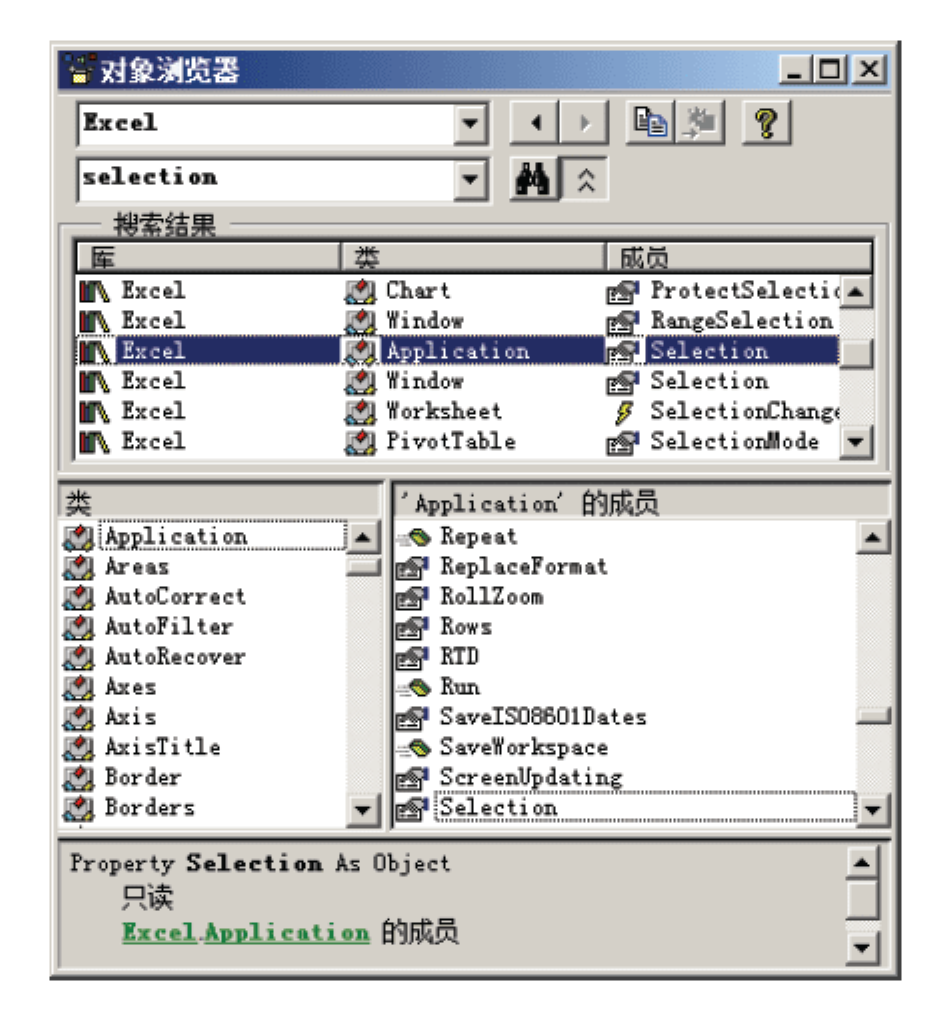


图 3.25　搜索对象



图 3.26　“帮助”窗口

提示： 在如图 3.25 所示的成员列表框中，图标表示属性，图标表示方法，图标表示事件，表示常数。

3.3　使用编辑器的代码输入功能

VBA 拥有大量的函数、对象和方法，用户要完全记住是不可能的。在 Visual Basic 编辑器中，所有代码的编写都是在“代码”窗口中完成的，Visual Basic 编辑器的“代码”窗

口的智能感应技术能够为开发者提供代码方面的帮助，使开发者快速而高效地完成应用程序的编写。

3.3.1　显示常用的属性和方法

Excel 包含很多的对象，每个对象又有各自的属性和方法，普通用户是很难记住所有的这些属性和方法的。在编写程序时，Visual Basic 编辑器序提供了能够使用户输入的代码自动提供对象属性和方法列表的功能，这样极大地方便了用户代码的输入。同样，在程序中输入常数后，VBA 同样能够给出常数列表，供用户选择需要使用的常数。下面通过具体的操作来介绍“属性/方法”列表和常数列表的使用方法。

（1）在“代码”窗口输入代码，当输入对象名和句点后，VBA 会自动给出一个下拉列表框，如图 3.27 所示。拖动列表框右侧的滚动条可以查看所有可用的属性和方法，双击需要的项目即可将其插入到程序中。如果在输入句点后继续输入属性或方法的前几个字母，VBA 会在列表自动找到匹配的项目，此时按 Enter 键即可将其插入程序，同时程序的输入将另起一行。

（2）Visual Basic 编辑器为代码的编写提供了一个“编辑”工具栏，这个工具栏在默认状态下是隐藏的。选择“视图”→“工具栏”→“编辑”命令可使“编辑”工具栏显示。在代码编写时，如果需要获得对象、属性或方法提示，可以单击“编辑”工具栏的“属性/方法”按钮打开一个列表框。在列表框中双击需要添加的内容即可将其直接添加到代码中，如图 3.28 所示。

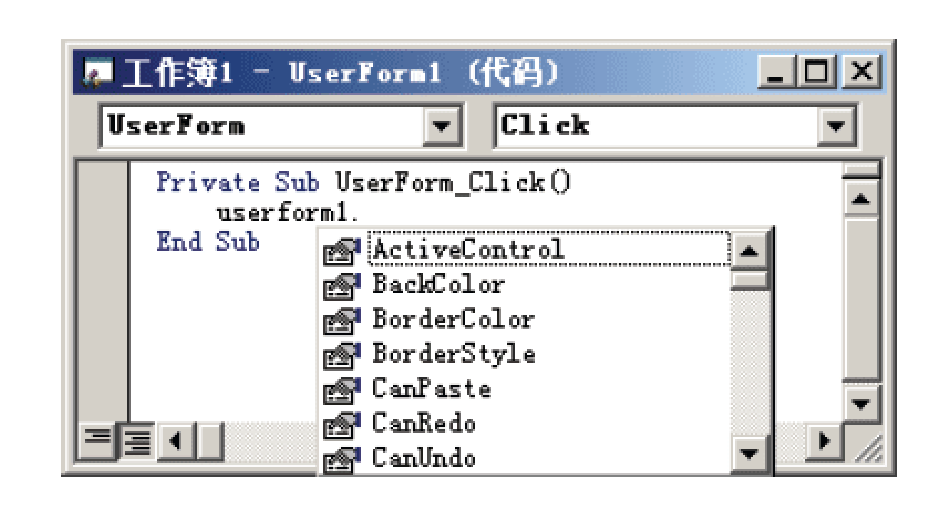


图 3.27　“属性/方法”列表

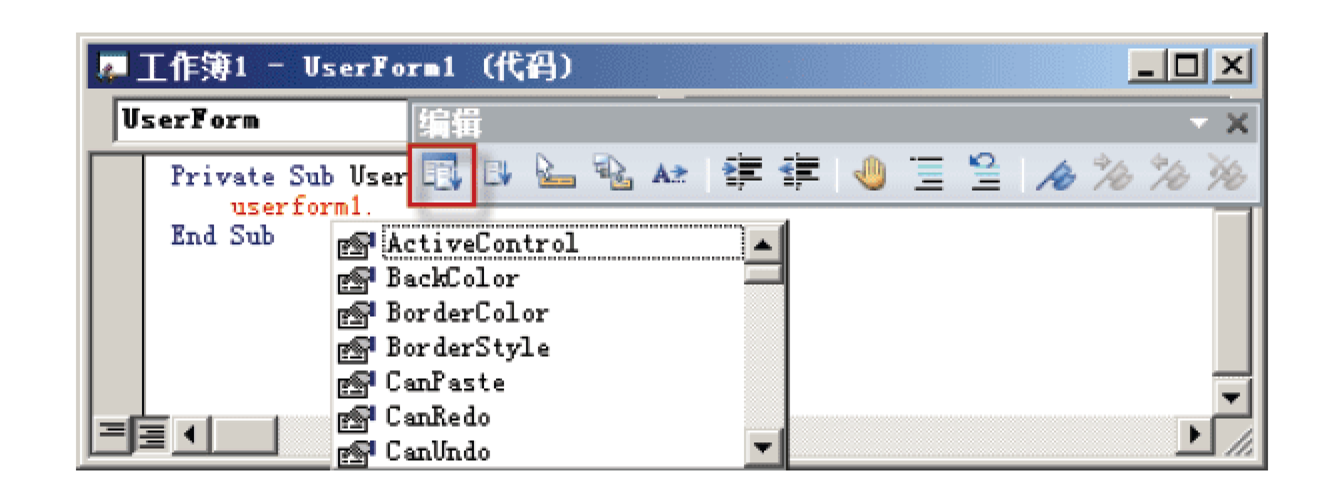


图 3.28　使用“编辑”工具栏打开“属性/方法”列表

提示： 在出现“属性/方法”列表后，按 Esc 键将取消该列表。以后再遇到相同的对象，列表也将不会再出现。此时，如果需要获得“属性/方法”列表，可以按 Ctrl+J 键。也可右击，然后在弹出的快捷菜单中选择“属性/方法列表”命令。

（3）在“代码”窗口中输入 VBA 常数后，如果输入“＝”，Visual Basic 编辑器会自动弹出一个“常数列表”列表框，如图 3.29 所示。双击列表中的选项，即可将其值输入代码中。

提示： 当显示“常数列表”后，可以使用键盘上的上下方向键选择列表中的选项，按空格将将选择内容输入程序。如果按 Esc 键，将关闭该列表。单击工具栏中的“常数列表”按钮或按 Ctrl+Shift+J 快捷键同样能够打开该列表。

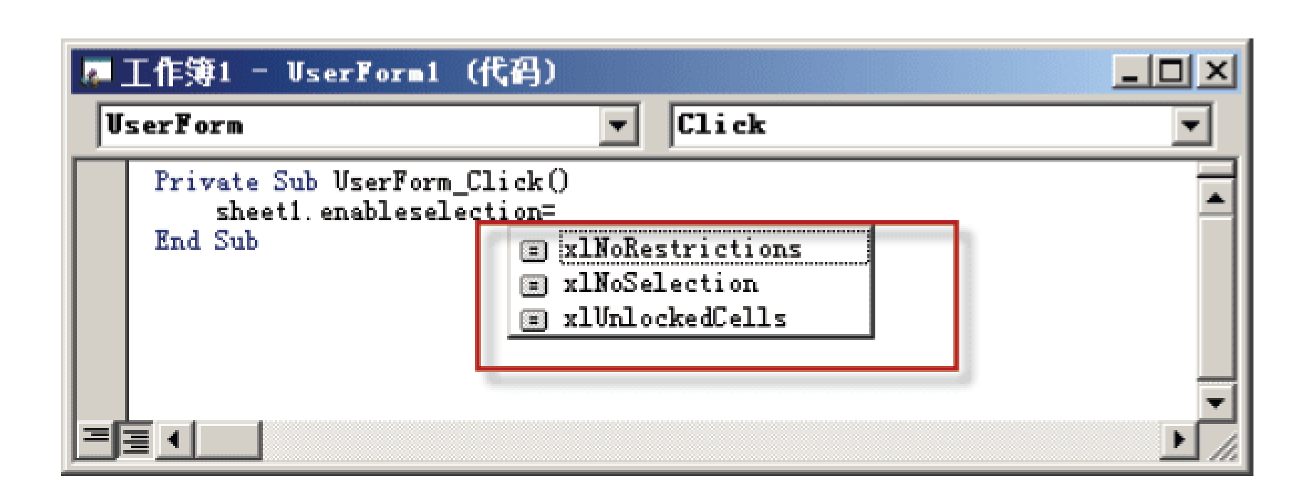


图 3.29　显示“常数列表”

3.3.2　显示参数

在编写程序时经常会用到 Excel 函数，这些函数在使用时往往需设置一个或多个参数。Visual Basic 编辑器为了方便函数的使用，提供了函数的参数信息提示功能。下面介绍该功能的使用方法。

在 Visual Basic 的“代码”窗口中输入 Excel 函数后，如果函数需要参数，在输入函数名和函数的左括号后，在光标下就会出现一个提示框。这个提示框将显示函数需要的参数，随着参数的输入，提示框会将当前需要输入的函数加粗显示，如图 3.30 所示。

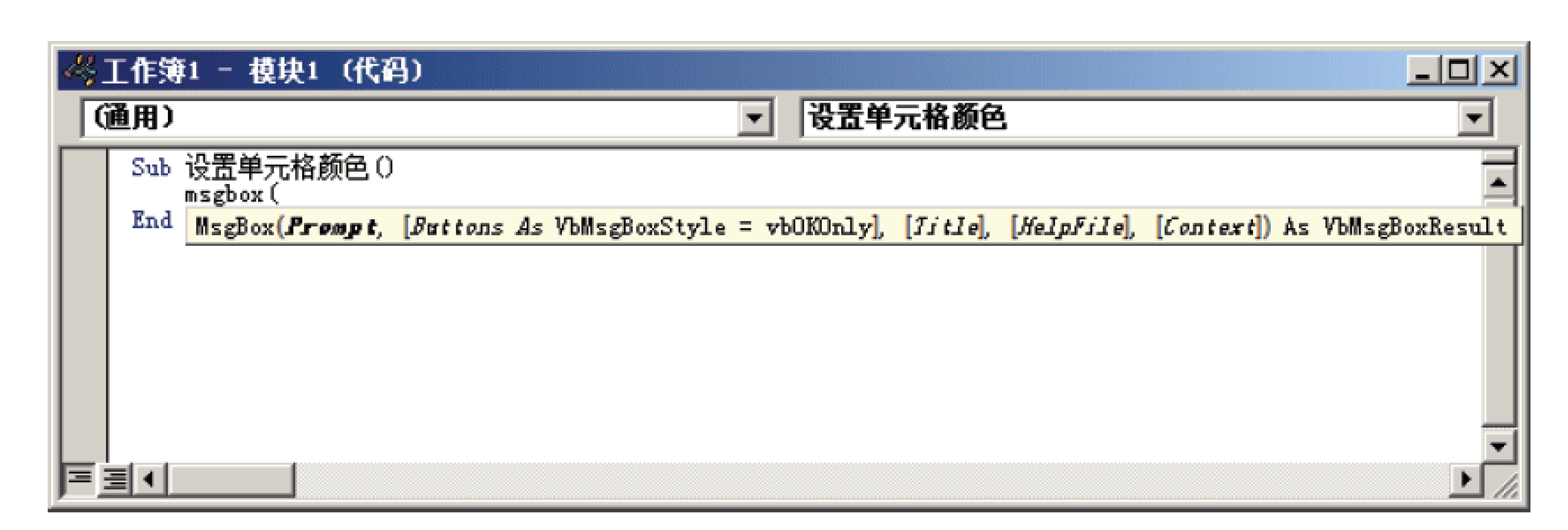


图 3.30　显示“参数提示”

3.3.3　使用快速信息

为了方便程序的编写，在“代码”窗口中输入关键字的前几个字母后，Visual Basic 会自动输入关键字后面的字母，这就是自动生成关键字功能。当在程序中输入 Visual Basic Application 指令、函数、方法、过程名或常数后，VBA 也能够将它们的值实时显示出来，这就是 Visual Basic 编辑器的快速信息功能。下面通过具体的操作来介绍这两种功能的使用方法。

（1）打开“编辑”工具栏，在“代码”窗口中输入一个关键字的前几个字母，单击工具栏上的“自动生成关键字”按钮 A≥，则关键字后面的字母将会自动输入。如果与输入字母相匹配的关键字有多个，则 Visual Basic 编辑器会给出一个下拉列表，用户可以从中选择需要的关键字，如图 3.31 所示。

提示：使用自动生成关键字功能能够提高代码编写的速度，同时可以有效地减少代码输入的错误。要自动生成关键字，也可以在输入字母后按“Ctrl+空格”键。

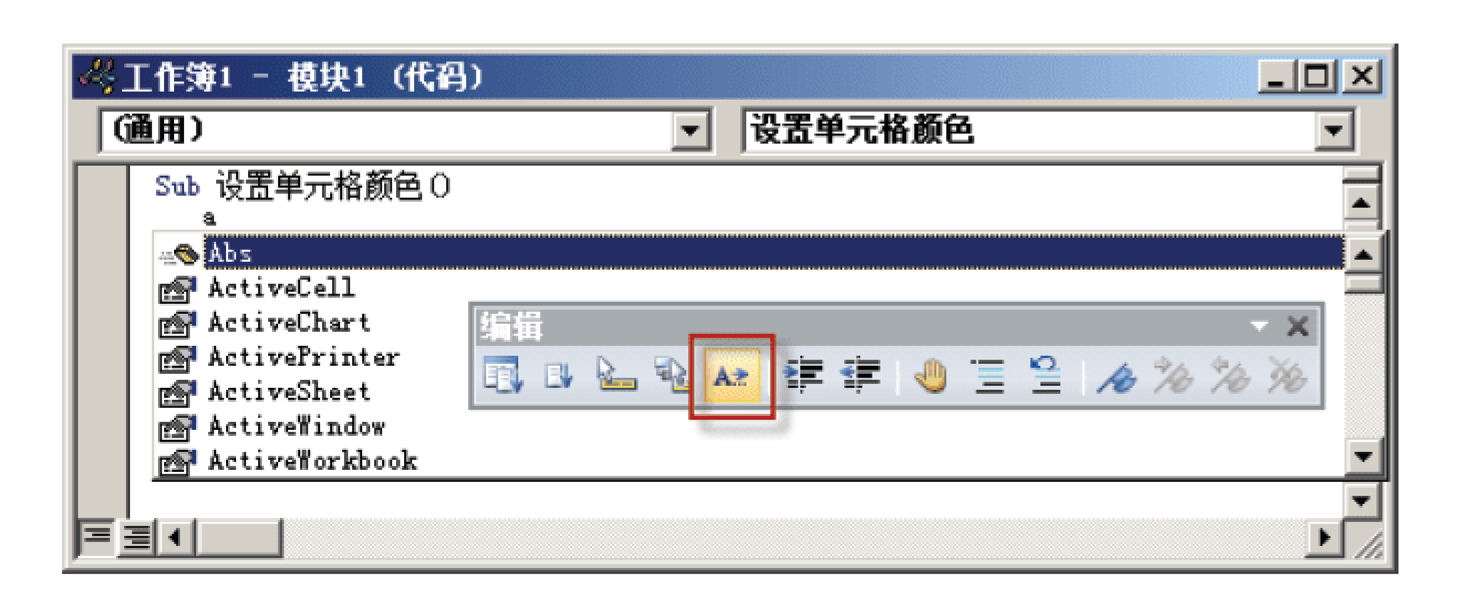


图 3.31　自动生成关键字

（2）在“代码”窗口中输入 VBA 指令、函数、方法、过程名或常数，单击“编辑”工具栏上的“快速信息”按钮，VBA 会显示该项目的语法或常数的值，如图 3.32 所示。

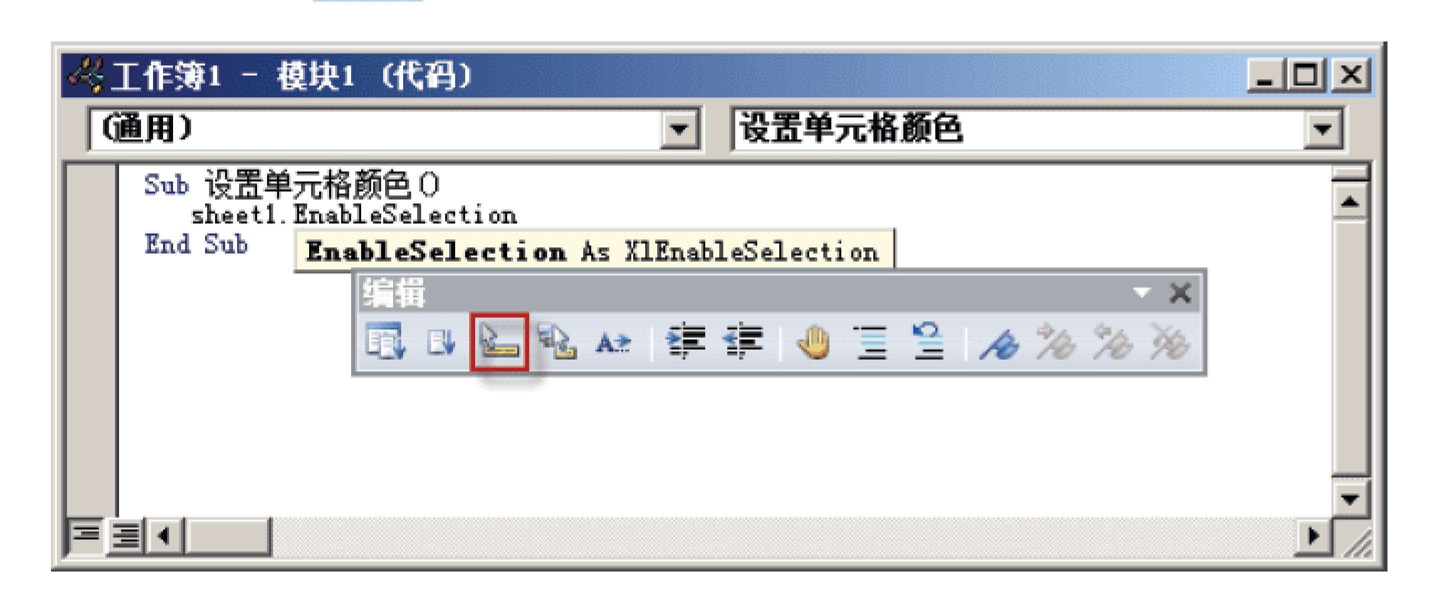


图 3.32　显示“快速信息”

注意： Visual Basicual 编辑器允许用户根据个人的需要来进行设置程序开发环境。对开发环境进行合理地设置，创建符合用户操作习惯的开发环境，能够在程序开发中提高工作效率和代码的正确率。Visual Basic 编程环境的设置是使用“选项”对话框来实现的，选择“工具”→“选项”命令可以打开该对话框，通过对该话框中的各个选项卡的设置能够设置本节中介绍的各项提示是否显示。

3.4　调试 VBA

对于应用程序的开发，程序的调试是一个重要的步骤。按设计要求编写的代码可能包含很多错误，这就需要通过调试来找出其中的错误并将其修复。本节将介绍在 Visual Basic 编辑器中调试 VBA 应用程序的知识。

VBA 程序有 3 种模式：设计时、运行时和中断模式。简单地说，设计时就是 VBA 代码编写状态，此时进行的是程序代码的输入或编写。运行时就是 VBA 程序处于运行的状态。在中断模式下，程序的运行将被暂停，用户能够检测当前的运行结果并对相关的变量的值进行修改。

程序设计时一般包括设计用户窗体和编写程序代码这两个过程。在设计时，可以设置断点并创建监视表达式，但不能运行代码或使用调试工具。此时，可以通过选择“运行”→“运行子程序/用户窗体”命令从设计时切换到运行时，使程序运行。如果需要切换到调试模式，可以选择“调试”→“逐语句”命令。

当程序处于运行时，应用程序处于运行状态，用户可以与运行的应用程序进行交互。在这种状态下，用户还可以查看程序代码，但不能对代码进行修改。要切换到设计时，可选择“运行”→“重新设置”命令。如果要切换到中断模式，可选择“运行”→“中断”命令。

在中断模式下，运行的应用程序处于暂停的状态。此时，可以查看程序代码或对程序代码进行编辑，并对操作数据进行检测或修改。同时，还可以重新运行程序，也可以结束程序运行或使程序从当前位置继续运行。在该模式下要切换到运行时，可以选择“运行”→“继续”命令。如果需要切换到设计时，可以选择“运行”→“重新设置”命令。

提示： 设计时、运行时和中断模式实际上是程序处于的 3 种状态。除了可以使用菜单命令来实现这 3 种状态键的切换之外，还可以使用工具栏按钮来进行切换。单击工具栏的 按钮可以使程序进入运行时，单击 按钮可暂停程序的运行，即进入中断模式。单击 按钮可切换到设计时，即停止当前程序的运行。

在程序设计时和中断模式下，可以设置断点。当程序运行到断点语句时，程序运行会中断。在对应用程序进行调试时，如果预计某段代码会出现问题，可以在该段代码前面设置断点，暂停代码的执行。然后通过逐语句执行来找出具体的问题代码。下面将介绍断点在程序调试中的使用方法。

（1）在“代码”窗口中找到需要设置断点的语句，将鼠标光标放置在代码行中。选择“调试”→“切换断点”命令设置断点，如图 3.33 所示。

（2）按 F5 键运行程序，程序运行到断点位置将暂停，同时标识出暂停的位置，如图 3.34 所示。

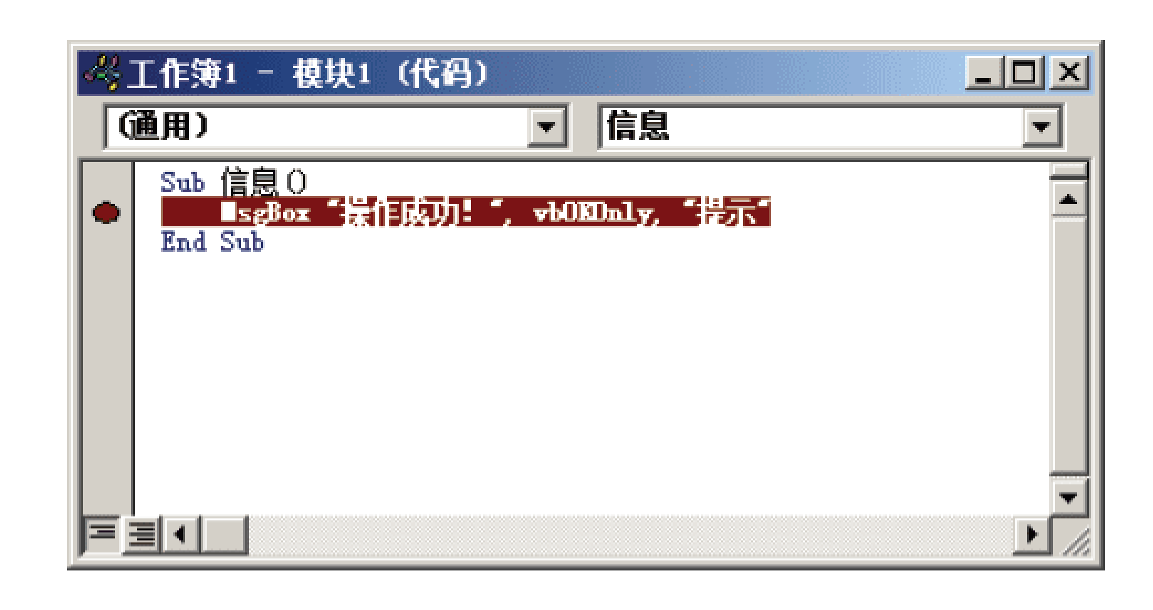


图 3.33　设置断点

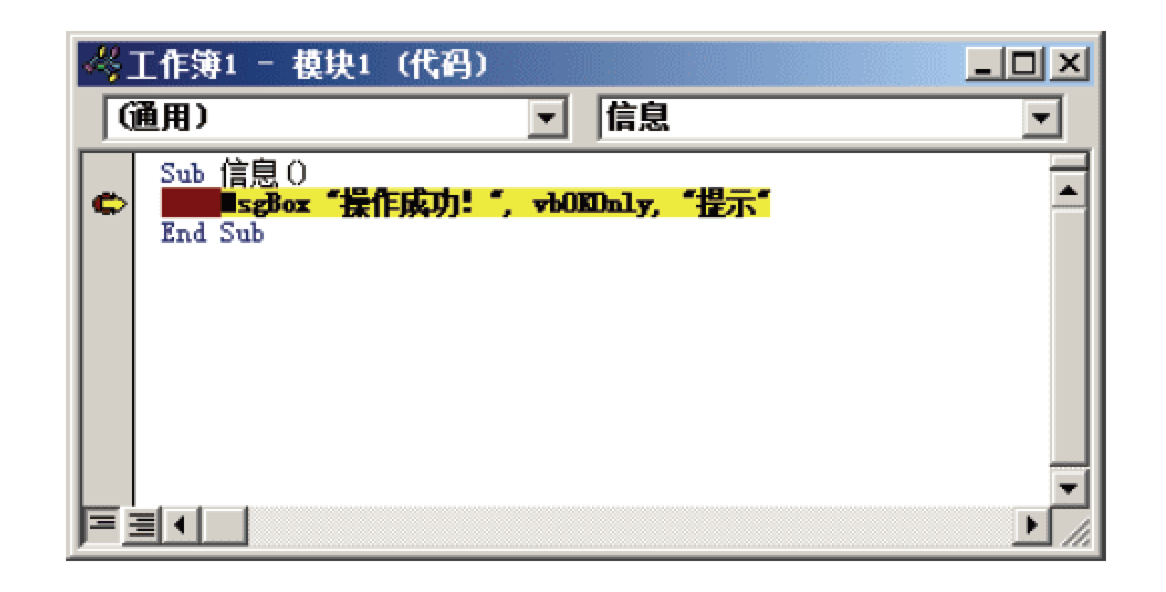


图 3.34　标示出暂停位置

提示： 实际上，设置断点最快捷的方法就是在“代码”窗口的边界标识条上单击。按“F9”键可以在输入点光标所在的代码段处设置断点，按“Ctrl+Shift+F9”键或选择“调试”→“清除断点”命令可以清除创建的断点。

（3）反复选择“调试”→“逐语句”命令或直接按“F8”键，程序将从断点开始逐条语句向下运行，同时在“代码”窗口中显示代码运行的位置，如图 3.35 所示。此时，能够很容易地找到出错的语句。

提示： 在找到出错语句后，可以按 F5 键继续执行过程中剩余的语句而不必再逐条运行程序了。同时，过程执行完后，断点不会被自动清除。但是当关闭工作簿后，设

置的所有断点将会被自动清除。另外，在对程序进行调试时，VBA 允许在一个过程中设置多个断点。

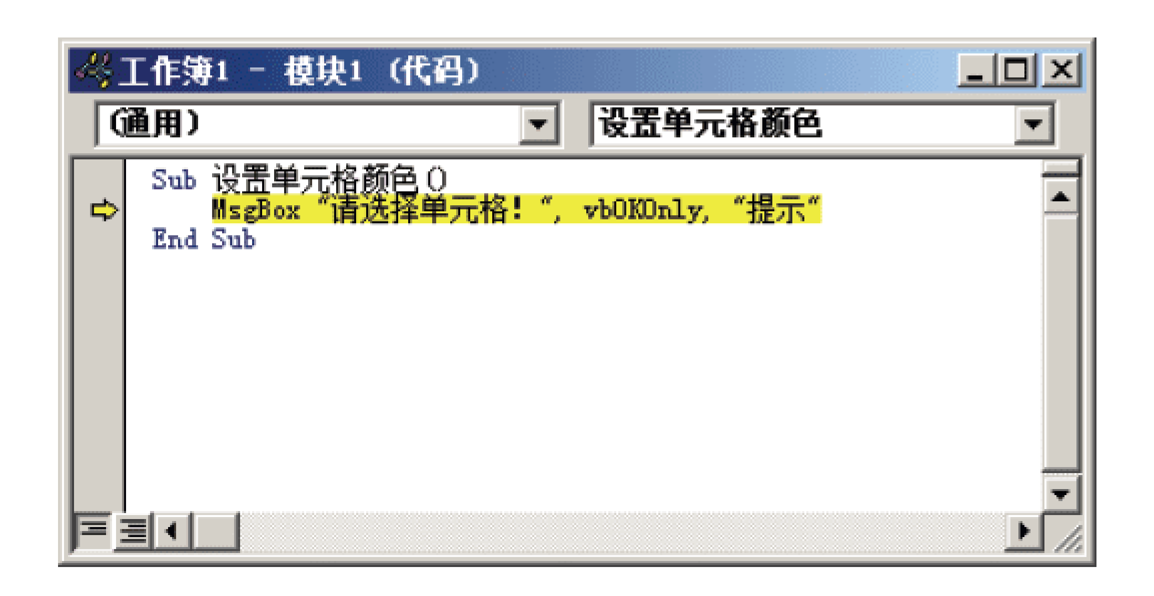


图 3.35　从断点向下逐条运行程序

3.5　小　　结

本章主要介绍了 Visual Basic 编辑器的构成元素、Visual Basic 编辑器中常用窗口的作用、快速输入程序代码的方法，以及定制 VBA 编辑器界面和使用断点调试 VBA 程序的方法。通过本章的学习，读者将能够对 Visual Basic 编辑器的作用有充分的认识，了解 VBA 编程的有关基础知识，为后面的学习打下基础。

Visual Basic 编辑器是一个强大的 VBA 编程工具，灵活地使用它将能够更快更准确地编写有效的程序代码。掌握 VBA 离不开不懈的学习。学习 VBA，除了充分地使用 Visual Basic 编辑器自带的帮助文档之外，还应该利用网络资源。合理地应用网络资源，可以找到需要的知识，获得先行者的宝贵经验。另外，借助于录制宏来学习 VBA 也是一个好的方法。希望读者们通过对宏代码的分析，能够熟悉各种模式的对象，掌握实现功能的方法。

3.6　本 章 习 题

1．使用下面哪个快捷键能够在 Excel 2010 中打开 Visual Basic 编辑器？（　　）

A．Alt+F11　　B．F1　　C．Ctrl+F10　　D．Alt+V

2．要移除一个已经创建的模块，应该在下面哪个窗口进行操作？（　　）

A．“代码”窗口　　B．“工程管理器”窗口

C．“属性”窗口　　D．“对象浏览器”窗口

3．在“代码”窗口中输入程序的过程中，下面哪个选项是 Visual Basic 编辑器不能自动显示的？（　　）

A．属性/方法列表　　B．常数列表

C．参数信息　　D．代码拼写错误提示

4．在输入程序代码时，如果需要显示函数的参数信息，应该在 VBA 编辑器的“选项”对话框的“编辑器”选项卡选中下面哪个复选框？（　　）

A．自动语法检测　　B．要求变量声明

C．自动列出成员　　　　　　　　　D．自动显示快速信息

5．Visual Basic 编辑器中，代码窗口的主要作用是什么？

6．Visual Basic 编辑器中，对象浏览器的主要作用是什么？

7．Visual Basic 编辑器中，立即窗口的主要作用是什么？

8．在 VBA 程序设计中，本地窗口的主要作用是什么？

9．在 Visual Basic 编辑器中，监视窗口的主要作用是什么？它与本地窗口的区别是什么？

10．编写程序在立即窗口中显示文字“Hello VBA！”。

第 4 章　VBA 变量和运算符

本章介绍 VBA 语言的基础知识，这是学习每种语言的第一步。变量和常量是程序中常用的构成元素，在编制程序的过程中，可以使用系统常量，也可以自定义常量。变量是用于保存程序运行中的临时变量的元素，同时介绍了运算符和表达式的使用方法。本章的主要内容和学习目标有：

- 掌握在程序中使用系统常量和用户自定义常量；
- 了解变量和常量的概念以及作用域和存活期的概念，掌握代码中变量和常量的声明方法；
- 掌握 VBA 常用的运算符和表达式，能够熟练地应用运算符书写各种表达式。

4.1　认 识 常 量

常量就是在程序运行过程中其值始终保持不变的量。常量就像是不变数据的容器，使用常量能够增强程序的可读性，如在程序中使用 Age 显然要比直接使用其值 36 要更好理解。在 VBA 中有两种形式的常量，它们是一般常量和符号常量。

4.1.1　定义系统常量

在 VBA 中，在程序代码中直接给出的数据即为一般常量，这种常量也可称为直接常量。符号常量是程序中使用符号来表示的常量。在 Excel 2010 中，符号常量可以分为两类，一类是系统内部的符号常量，即系统常数。另一类是用户自定义符号常量。这里所说的系统常量实际上指的是两类系统常量，一类是 VBA 系统内部的符号常量，例如，定义提示对话框样式的常量 vbOkOnly 和定义日期的常量 vbSaturday 等。另一类是 Excel 系统内部的符号常量，例如，用于设置工作表显示状态属性的常量 xlSheetVisible。

系统常量名采用的是大小写混合书写的模式，其前缀表示定义的常量的对象库名。VBA 系统常量的前缀是小写的 vb，而 Excel 系统常量名前通常用小写的 xl 作为前缀。对于这两类常量，都可以使用 Visual Basic 编辑器的帮助文档或使用对象浏览器来查阅某个系统常量的名称以及具体的值。

【范例 4-1】 使用系统常量 xlToLeft 实现在工作表中按 Enter 键将活动单元格左移的功能，代码如下所示。

```
Sub 活动单元格左移()
    Application.MoveAfterReturn = True      '设置属性值为 True 允许移动
    Application.MoveAfterReturnDirection = xlToLeft '设置向左为移动方向
```

```
04    End Sub
```

【运行结果】在 Excel 2010 中运行名为“活动单元格左移”的宏，在工作表中按 Enter 键，活动单元格左移，如图 4.1 所示。

【代码解析】示例展示了系统常量的使用方法。在第 03 行的代码中 xlToLeft 是一个系统常数，对象的 MoveAfterReturnDirection 属性值被设置为该系统常量。要保证在按 Enter 键后活动单元格左移，必须将 MoveAfterReturn 属性设置为常量 True。

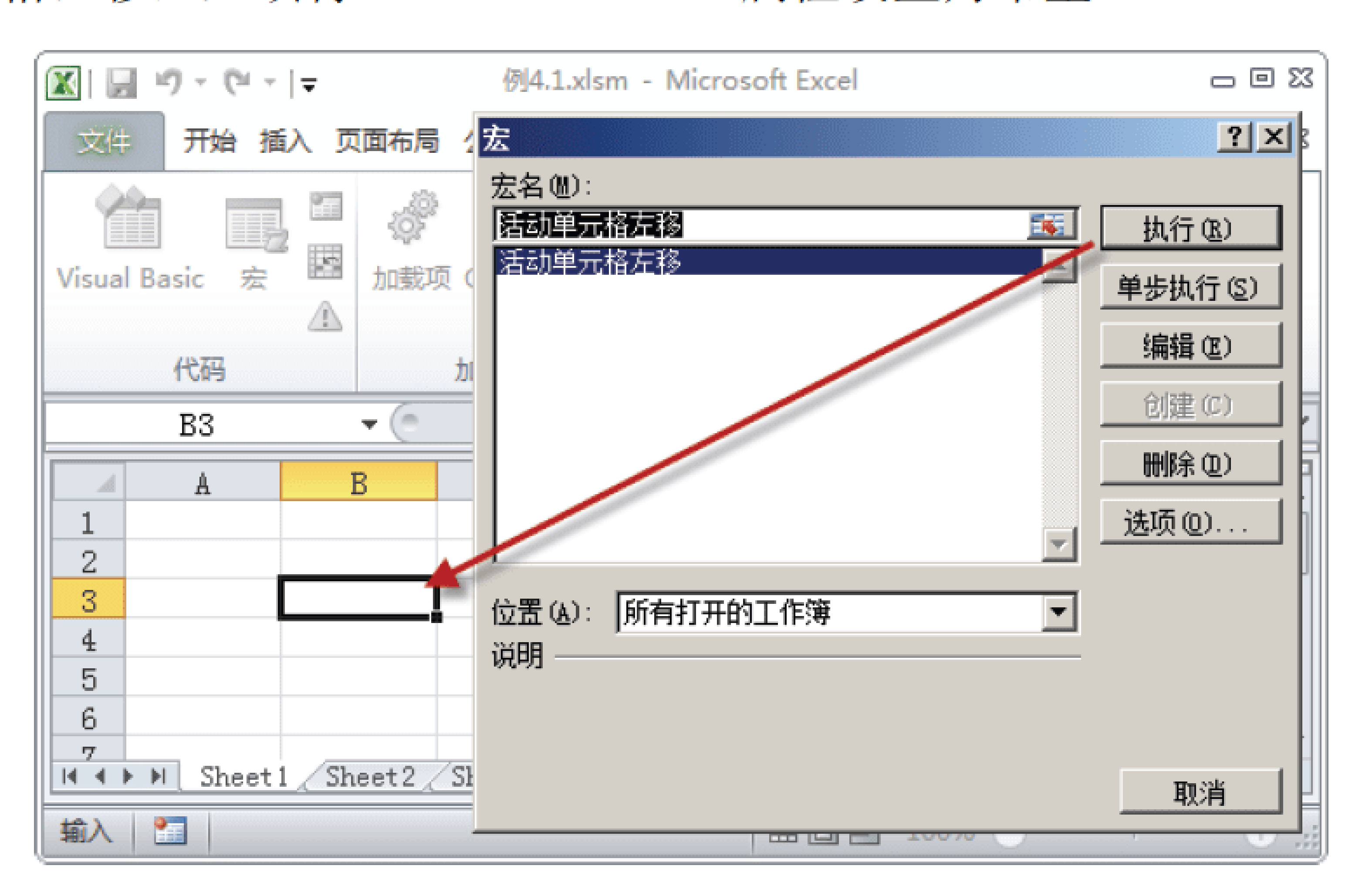


图 4.1　执行宏并按 Enter 键使单元格左移

4.1.2　自定义常量

所谓的自定义常量是相对于系统常量而言的，它是由用户在程序中定义的常量。在 VBA 程序中，用户可以使用 Const 语句来声明自定义常量，没有进行声明的常量是不能使用的。使用 Const 语句声明变量的语法格式如下：

```
[Public|Private] Const 常量名 [As 数据类型]=表达式
```

参数说明如下所示。

- Public：表示公用常量，用于在模块中声明符号常量，常量将在整个应用程序中使用。
- Private：表示私有常量，即声明所使用的常量只能在所声明的过程中使用。如果省略了 Public 或 Private，则默认为 Private。
- Const：定义符号常量关键字。
- 常量名：声明的常量的名称。
- 数据类型：声明常量的数据类型。
- 表达式：可以是数字、字符串，也可以是结果为数字或字符串或布尔值的表达式。

注意：程序中的数据是放置于内存中的，在 VBA 中，可以使用特定名称来表示数据在内存的位置，这个名称就是标识符。通俗地说，标识符实际上是 VBA 程序中某个变量和常量的名称。

作为标识符的名称不能和 VBA 本身的过程、语句和方法同名。在命名标识符时，不能在范围相同的层次中使用重复的名称。标识符中不能使用空格、句点（.）、感叹号（！）或者诸如@、#、$和&等符号，且不能超过 255 个字符。

【范例 4-2】 编写程序，使用系统常量填写“职工信息表”，代码如下所示。

```
01  Public Const Code = "010103"                    '声明“工号”常量
02  Public Const Age = 21                           '声明“年龄”常量
03  Public Const Name = "郭轶凡"                     '声明“姓名”常量
04  Public Const Sex = "男"                          '声明“性别”常量
05  Sub 声明自定义常量()
06      Cells(3, 1) = Code                          '向单元格中填入工号
07      Cells(3, 2) = Name                          '向单元格中填入姓名
08      Cells(3, 3) = Sex                           '向单元格中填入性别
09      Cells(3, 4) = Age                           '向单元格中填入年龄
10  End Sub
```

【运行结果】在 Visual Basic 编辑器的工程资源管理器中右击，在弹出的快捷菜单中选择“插入”→“模块”命令，插入一个模块。在该模块的“代码”窗口中输入上述程序代码，按 F5 键运行程序。程序中常量的值被输入到当前工作表的指定单元格中，如图 4.2 所示。

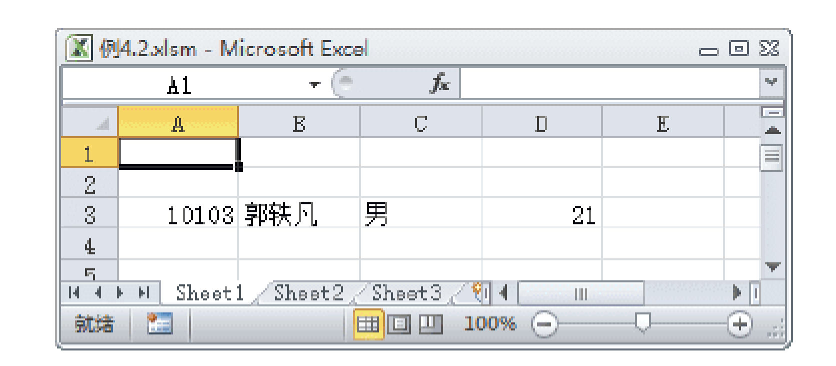


图 4.2　程序运行的效果

【代码解析】本段代码演示自定义常量的定义和使用方法。在程序的第 01~04 行声明了 4 个自定义共有常量，在代码的第 06~09 行，将自定义常量的值填入当前工作表的指定单元格中。

提示： 在程序中，常常使用大写字母来给常量命名，这样能够将其与变量区分开，增强程序的可读性。

4.2　使 用 变 量

在程序设计中，变量相当于一个存储数据的容器，用于存储程序运行时产生的临时值。根据数据类型的不同，变量也分为不同的类型。与常量不同，变量的值在程序运行时是可以根据需要随时修改的。本节将介绍在 VBA 程序中变量的使用方法。

4.2.1　在 VBA 中声明变量

变量在使用前需要声明，声明即是在系统中为数据分配存储单元。变量声明后，每个

变量拥有一个名字，并且具有一个数据类型。在 VBA 中，可以使用 Dim 语句来对变量进行声明，其具体的语法格式如下：

```
Dim 变量名[As 数据类型]
```

参数说明如下所示。

- Dim：关键字，用于变量声明。
- 变量名：声明的变量的名称。
- As 数据类型：指定变量的数据类型。

提示： VBA 的数据类型包括数值型、字符型、布尔型、日期型、货币型和对象型。其中，数值型数据可以分为整型、长整型、字节型、单精度浮点型和双精度浮点型等。各个数据类型的具体含义以及使用方法可以在 VBA 帮助文档中查阅，这里不再赘述。

下面看看变量声明的几个实例。

- 声明一个字符串变量：

```
Dim name As String
```

- 在一个语句中可以同时声明几个变量，此时应该将所有变量的数据类型都包含进去。

```
Dim x As Integer,y As Integer,z As Integer
```

注意： 如果将上面的声明语句改为下面的形式，则变量 x 和 y 的数据类型将是 Variant，只有 z 是 Integer 类型。这是在需要将多个变量同时定义为相同的数据类型时应该注意的。

```
Dim x ,y z As Integer
```

- 将对象变量声明为工作表类型。

```
Dim s As WorkSheet
```

【范例 4-3】 编写程序，使用变量来填写“商品结算单”，代码如下所示。

```
Sub 变量使用()
    Dim n As String, c As Integer, d As Currency, z As Currency, r As Date    '声明变量
    n = "联想笔记本电脑"                    '品名赋予变量
    c = 3                                   '产品数量赋予变量
    d = 4900                                '产品单价赋予变量
    z = c * d                               '计算总价被赋予变量
     r = #10/5/2008#                        '结算日期赋予变量
    Cells(3, 1) = n                         '填入品名
    Cells(3, 2) = c                         '填入产品数量
    Cells(3, 3) = d                         '填入产品单价
    Cells(3, 4) = z                         '填入产品总价
    Cells(3, 5) = r                         '填入结算日期
End Sub
```

【运行结果】在 Visual Basic 编辑器的工程资源管理器中右击，在弹出的快捷菜单中选择“插入”→“模块”命令插入一个模块。在该模块的“代码”窗口中输入上述程序代码，按 F5 键运行程序。程序中变量的值被输入到当前工作表的指定单元格中，如图 4.3 所示。

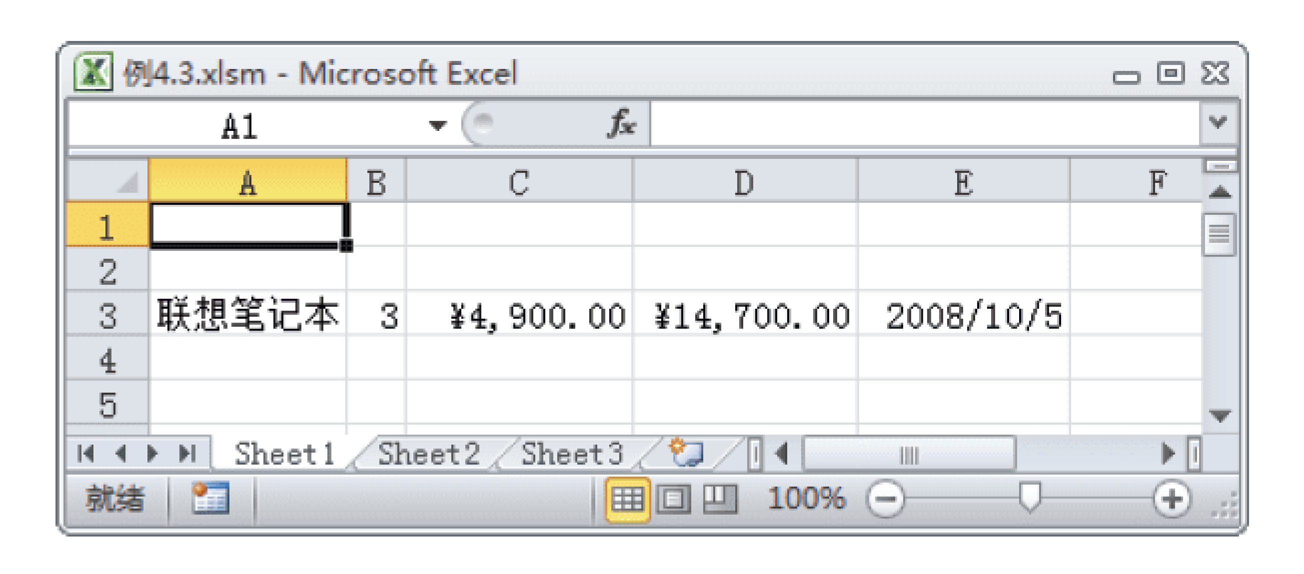


图 4.3　程序运行的效果

【代码解析】本段代码演示程序中变量的定义和使用方法。在程序的第 02 行对变量进行声明，声明时指明了变量的数据类型。程序的第 03~07 行将值赋予变量，第 08~12 行将变量值填入指定单元格中。

提示：对于日期型变量，日期字符必须使用数字符号“#”括起来，否则填入单元格中将得不到正确的日期值。

4.2.2　VBA 强制声明变量

在 VBA 中是可以使用未经声明的变量的，这种变量称为隐式变量。对于隐式变量系统会自动将其定义为 Variant 数据类型。使用隐式变量时，如果在过程中错误地拼写了变量名，VBA 将不会有提示信息。由于使用的是 Variant 数据类型，程序将占用更多的内存。正是基于以上的两点原因，在程序中应该避免使用隐式变量，此时就需要强制声明变量。所谓的强制声明，就是在编写程序时必须要声明变量后变量才能使用。在 VBA 编辑器中通过有关的设置可以实现对变量的强制声明，设置方法如下。

（1）在 Visual Basic 编辑器中选择“工具”→“选项”命令打开“选项”对话框。选择“编辑器”选项卡，再选中“要求变量声明”复选框，如图 4.4 所示。

（2）单击“确定”按钮关闭“选项”对话框。当每次添加模块时，在模块的“代码”窗口中将会自动添加 Option Explicit 语句，如图 4.5 所示。

提示：Option Explicit 语句用于变量的强制声明。添加该语句后，如果遇到没有声明的变量，VBA 会认为是语法错误，将会给出错误提示，这样可以避免因没有声明变量而增加程序调试的复杂性。当然，这个语句也可以手动输入。

4.2.3　VBA 变量的作用域

不同的变量在 VBA 的过程中会有不同的作用范围，变量的作用域指的就是变量的有效作用范围。在使用 Dim 对变量进行声明时，Dim 在模块中的位置决定了变量的作用域。

在 VBA 中，变量根据作用域分为 3 个级别，依次是过程级变量、模块级变量和工程级变量。下面分别对这 3 个级别的变量进行介绍。

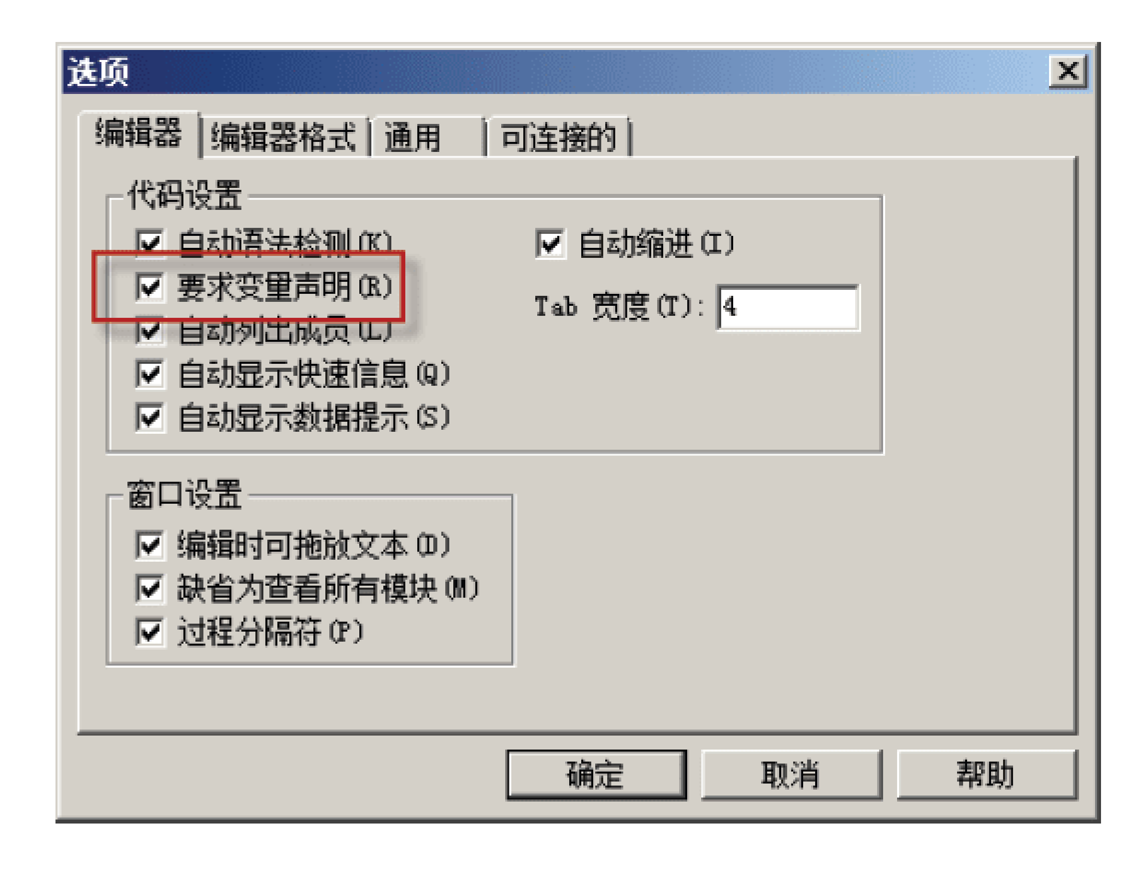




图 4.4 选中“要求变量声明”复选框

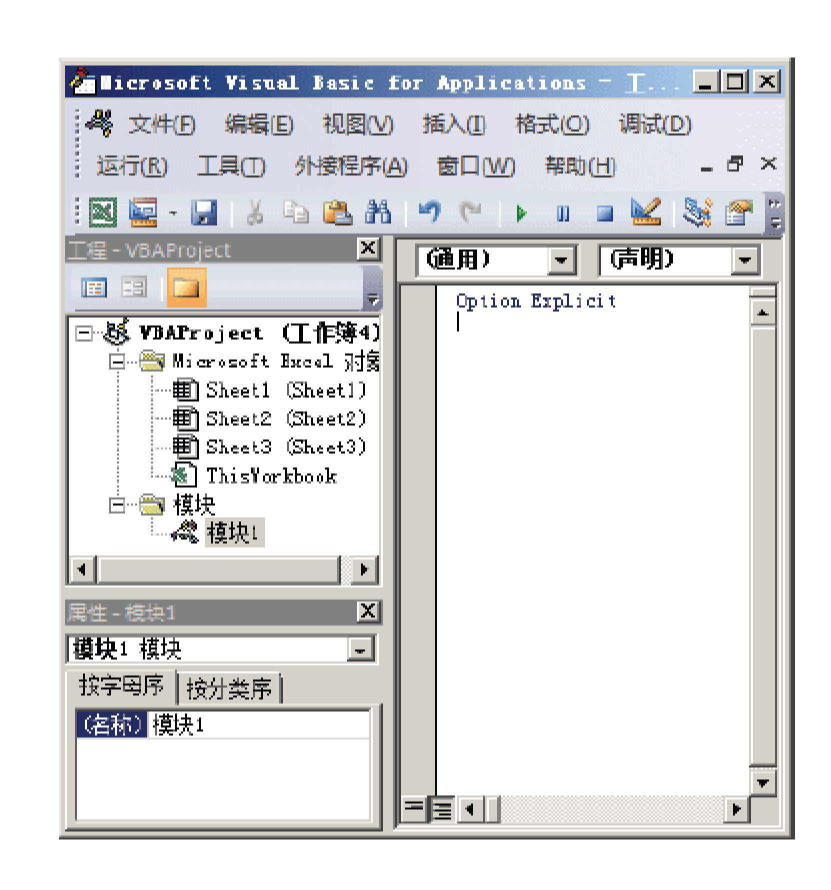


图 4.5 自动添加 Option Explicit 语句

- 过程级变量：过程级变量也称为局部变量。变量在过程中进行声明，只在此过程中使用。在过程结束后过程级变量将立即消失，VBA 将释放变量占用的内存空间。使用过程级变量将节省内存空间，并且相同的变量名能够在不同的过程中使用。
- 模块级变量：当需要在模块的多个过程间共享某个变量值时，需要使用模块级变量。模块级变量在定义时必须将变量的声明语句放置于模块的声明部分，即不能包括在任何过程中。
- 工程级变量：工程级变量也称为全局变量，这种变量能够在工程的所有模块中被访问。在模块的声明部分用 Public 关键字声明的变量就是工程级变量。

注意：在模块中使用 Private 关键字生的变量是模块级变量。在程序中，应该尽量使用局部变量。只有当确实需要在不同过程间共享数据时才使用模块级变量。同样，对于工程级变量的数量也应该有所控制。

【范例 4-4】 定义过程级变量和模块级变量，使用“立即窗口”显示变量的值，比较不同的运行结果，代码如下所示。

```
Option Explicit                          '强制声明变量
 Sub 测试1()
     Dim gc As String                    '定义一个过程级变量
     gc = "过程级变量"                   '给过程级变量赋值
     mk = "模块级变量"                   '给模块级变量赋值
     Debug.Print  mk                     '显示模块级变量值
     Debug.print  gc                     '显示过程级变量的值
End Sub
Sub 测试2()
     gc="过程级变量"                     '给过程级变量赋值
     mk="模块级变量"                     '给模块级变量赋值
     Debug.Print  mk                     '显示模块级变量的值
     Debug.Print  gc                     '显示过程级变量的值
End Sub
```

【运行结果】运行“测试 1”过程时，“立即窗口”显示的效果如图 4.6 所示。运行“测

试 2”过程时，Visual Basic 编辑器给出错误提示，同时表示出未定义的变量，如图 4.7 所示。



图 4.6　“测试 1”过程的运行效果

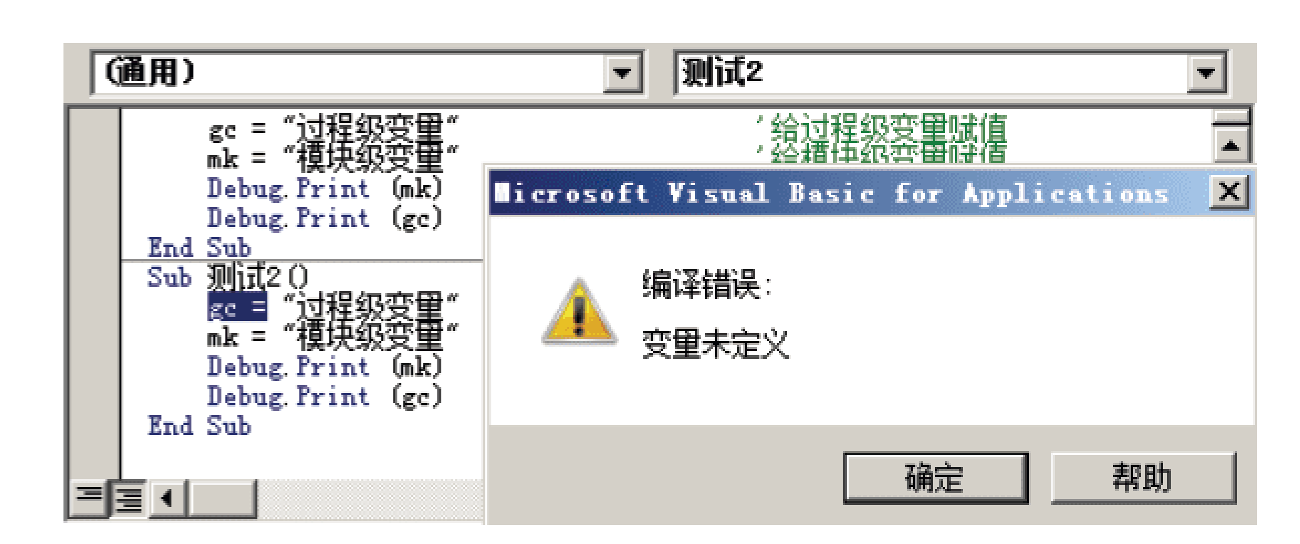


图 4.7　“测试 2”过程的运行结果

【代码解析】这段程序代码将演示过程级变量和模块级变量在作用域上的不同。程序的第 01 行代码强制变量声明，如果删除该语句，则“测试 2”过程在运行时将不会给出错误提示，此时“立即窗口”显示的结果如图 4.8 所示。比较这两个过程运行后的结果可以看到，模块级变量的值在运行这两个过程时都显示，过程级变量的值只在定义了该变量的过程运行时才能显示。



图 4.8　取消强制变量声明语句后的显示效果

4.2.4　详解 VBA 变量的生存周期

变量的生存周期指的是变量能够保存其值的时间。根据变量生存周期的不同，变量可以分为动态变量和静态变量两类。

在程序运行到变量所在的过程时，变量被分配内存单元。当退出该过程，变量使用的内存单元会自动释放，其值也就消失了。当再次进入该过程时，如果变量被重新初始化，这样的变量就是动态变量。如果程序在退出变量所在的过程时，变量值仍然保留在内存中，即变量的内存空间不释放。当程序再次进入该过程，变量值能够继续使用，这样的变量就是静态变量。

使用 Dim 语句声明的变量都是动态变量，而使用 Static 语句可以将变量声明为静态变量，如下面的语句就定义了一个字符串型的静态变量 name：

```
Static name As String
```

【范例 4-5】 分别定义静态变量和动态变量，比较它们的生存周期，代码如下所示。

```
Sub 静态变量和动态变量()
    Dim a As Long                         '定义动态变量'
    Static b As Long                      '定义静态变量'
    a = a + 2                             '变量值增加 2'
    b = b + 2
    Debug.Print  a                        '显示变量 a 的值'
```

```
07      Debug.Print  b                          '显示变量 b 的值'
08   End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口中输入示例代码。按 F5 键运行此段程序。将该模块运行 7 次，在“立即窗口”中可以看到每次代码运行后 2 个变量的值，如图 4.9 所示。

【代码解析】在代码中，变量 a 是动态变量，每次该过程运行时，a 都将被初始化，因此不管代码运行多少次，它的值都是 2（初始值 0 加上 2）。变量 b 是静态变量，其值只在第一次进入过程时初始化，而退出过程时其值保留在内存中。在每次进入运行该过程时，变量 b 都保持了原来的值，因此在“立即窗口”中显示的 b 的值是上一次的运行结果增加 2。

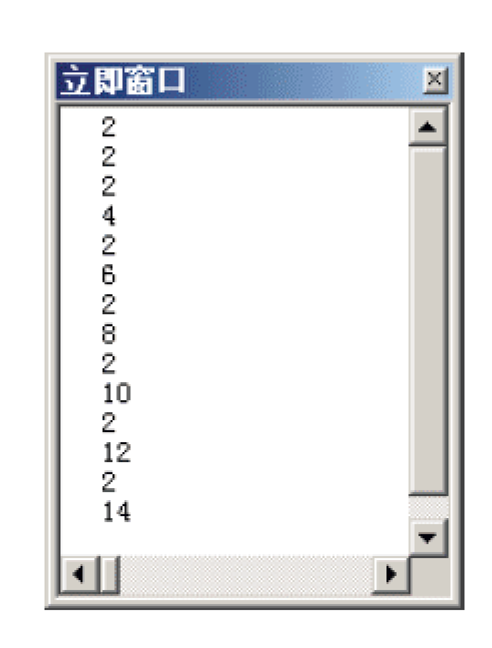


图 4.9　多次运行程序后的效果

注意：无论是哪种类型的变量，当 Excel 文档关闭时，变量占用的内存都将释放，变量及变量的值都将随之消失。

4.3　使用运算符和表达式

程序的运行少不了数据的运算，运算通过运算符来实现。运算符就是介于操作数之间的运算符号，实际上就是能完成特定运算的操作符。将常量、变量和函数等用运算符连接起来的运算式就称为表达式。VBA 提供了 4 种基本运算，它们是算术运算、比较运算、逻辑运算和连接运算。下面对相关的运算符进行介绍。

4.3.1　算术运算符与算术表达式

算术运算符是最为常见的运算符，用于在程序中进行数据的算术运算，如对两个数字进行加、减、乘、除等计算，以获得计算结果。VBA 常见的算术运算符包括：“+”、“-”、“*”以及“/”运算符，它们可以用于进行数据的加、减和乘除运算。另外，使用“^”运算符可以进行幂运算，“Mod”运算符计算两个数据相除后商的余数部分。

如果表达式是一个包含多个运算符的混合运算，那将涉及到运算的顺序问题。运算的顺序由运算符的优先级来决定。对于算术运算符的优先级，具有与传统数学运算相同的优先级顺序，即最高优先级的是乘方运算，其次是乘除运算，最后是加减运算。VBA 中的整数除法运算（\）的优先级要低于除法运算（/），求模运算（Mod）的优先级低于整数除法运算（\）的优先级，但这两个运算的优先级都高于加减运算。在上述介绍的运算中，优先级最低的是连接运算符（&）。

提示：&运算符可以进行连接运算，即强制 2 个表达式作为文字符串连接。例如，表达式"Excel" & "2010" 表示连接这两个字符串获得一个新的字符串，其结果为

"Excel2010"。而“\”运算符用来对两个数进行除法运算，但返回的结果是商的整数部分，不保留小数部分。如 10\3 的结果是 3。

【范例 4-6】 求一元二次方程 $x^2+4x+3=0$ 的根，代码如下所示。

```
01  Sub 求固定系数的一元二次方程的根()
02      Dim a As Integer, b As Integer, c As Integer    '定义二次项系数
03      Dim x1 As Double, x2 As Double                  '定义方程 2 根
04      Dim msg As String, title As String              '定义信息变量
05      a = 1                                           '二次项系数
06      b = 4                                           '一次项系数
07      c = 3                                           '常数项的值
08      x1 = (-b + (Sqr(b ^ 2 - 4 * a * c)) / (2 * a)  '求第一个根
09      x2 = (-b - (Sqr(b ^ 2 - 4 * a * c)) / (2 * a)  '求第二个根
10      title = "固定系数的一元二次方程求根示例"              '定义标题
11      msg = "二次项系数为" & a & ", 一次项系数为" & b & _
12      ", 常数项为" & c & "的一元二次方程的根为: " & Chr(10) & x1 & Chr(10) & x2
        '连接显示的信息'
13      MsgBox msg, vbOKOnly, title                     '显示计算结果
14  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口中输入示例代码。按 F5 键运行此段程序。在弹出的对话框中显示计算结果，如图 4.10 所示。

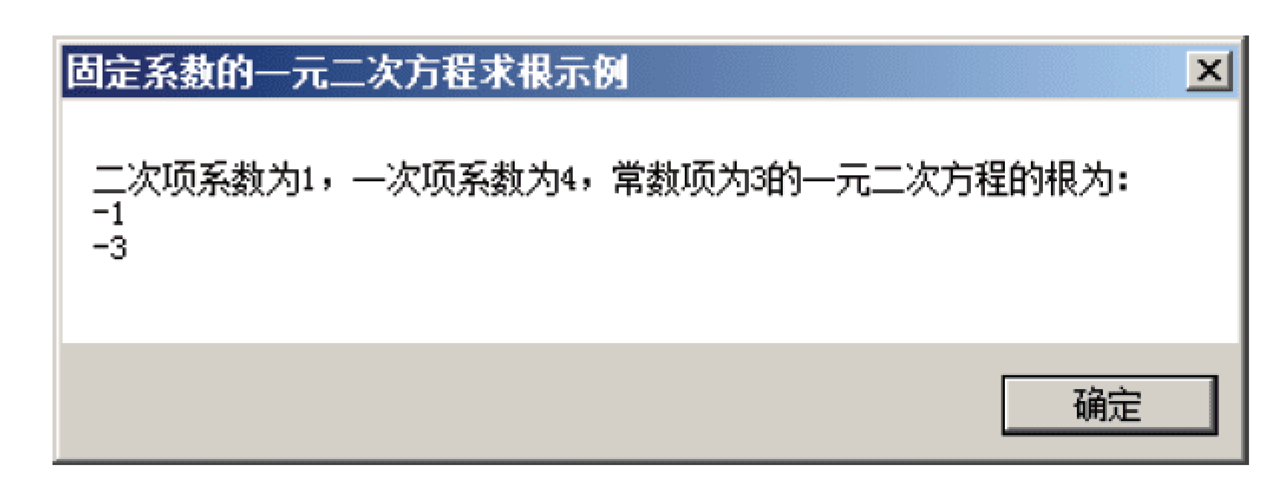


图 4.10　程序运行效果

【代码解析】这段代码用于求一元二次方程 $x^2+4x+3=0$ 的根。代码首先声明程序需要的变量，然后对变量赋值。在第 08 行和第 09 行程序代码中，使用 Sqr 函数来求数值的平方根，使用括号来改变运算顺序。在第 11 行程序代码中，使用连接运算符&来连接文字信息，使用 Chr(10）语句来实现文字显示的换行。

4.3.2　比较运算符与比较表达式

比较运算符用来表示两个或多个数值间的关系，其包括小于（<）、大于（>）、小于等于（<=）、大于等于（>=）、不等于（<>）、等于（=）、字符串匹配（Like）和对象引用比较（Is）。它们的结果只能是 True 或 False。关系运算符在使用时，如果需要将结果赋予某个变量，可以采用下面的格式：

```
运算结果＝表达式 1  关系运算符  表达式 2
```

比较运算符同样有它的优先级，其优先级按照由高到低的排列顺序为：等于（=）、不等于（<>）、小于(<)、大于（>）、小于等于（<=）和大于等于（>=）。而字符串匹

配（Like）和引用比较（Is）的优先级最低。

提示： Is 用于对两个引用对象进行比较，如果两个引用对象相同，则返回值为 True，否则为 False。Like 对 2 个字符串进行比较，如果字符串相匹配，则返回值为 True，否则为 False。

【范例 4-7】 比较两个表达式的大小关系，代码如下所示。

```
01  Sub 比较运算符的使用()
02      Dim a As Double, b As Boolean           '声明变量
03      b = (4 ^ 2 - 4 * 3 * 2) > 0             '比较数值大小
04      Debug.Print  b                          '在“立即窗口”中显示计算结果
05   End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口中输入示例代码。按 F5 键运行此段程序。在“立即窗口”中显示 b 的值，如图 4.11 所示。

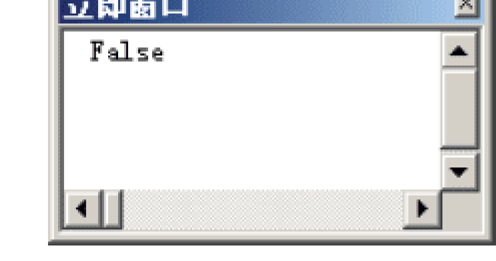


图 4.11　程序运行的结果

【代码解析】本示例用于演示比较运算符的作用。比较运算符的返回值是布尔型，在第 02 行代码中将变量 b 声明为该类型。第 03 行代码比较表达式“4 ^ 2 - 4 * 3 * 2”的值与 0 的大小，如果其值大于 0，则结果为 True，否则结果为 False，比较结果赋予变量 b。

4.3.3　逻辑运算符与逻辑表达式

逻辑运算符是连接表达式进行逻辑运算的运算符，其运算结果只有 True 和 False 这两种。VBA 包括 5 个常用的逻辑运算符，如 Not 运算符用于对表达式进行逻辑否运算，And 运算符用于两个表达式进行逻辑与运算，Or 运算符用于两个表达式的逻辑或运算。

【范例 4-8】 查看逻辑变量 a 和 b 蕴含运算的值，代码如下所示。

```
01  Sub 查看逻辑运算结果()
02      Dim a As Boolean, b As Boolean, c As Boolean     '声明逻辑变量
03      a = True                                         '变量赋值
04      b = False
05      c = a Imp b                                      '变量进行蕴含运算
06      Debug.Print  c                                   '显示计算结果
07   End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口中输入示例代码。按 F5 键运行此段程序。在“立即窗口”中显示 c 的值，如图 4.12 所示。

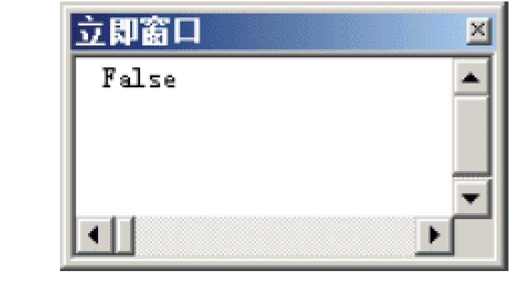


图 4.12　程序运行的结果

【代码解析】本示例演示逻辑运算的结果。本示例运行的结果都是逻辑型数据，所有变量声明为逻辑变量。读者可以在操作数值不变的情况下，更改第 05 行代码的操作符查看计算结果的变化。同时也可以更改操作数 a 和 b 的值，查看不同操作数在相同操作符下的运算结果。这样有利于读者加深对逻辑运算的理解。

提示： Imp 表示逻辑蕴含关系，读者想要了解其运算结果，可以更改示例中变量 a 和 b 的值，运行程序查看不同的运算结果。

4.4 小　　结

VBA 是 Visual Basic 的应用程序版本，是 Office 办公软件各套件的内嵌的编程语言。VBA 继承了 Visual Basic 语言的简单易用的特点，具有和 Visual Basic 相同的语法结构。通过在 Excel 中使用 VBA 编程，能够方便地对文档进行控制，使文档具有更为强大的功能，为开发人员对 Excel 的二次开发提供了更为方便快捷的手段。

学习 VBA 必须掌握其基本的语法规范和相关概念。本章介绍了 VBA 的常见数据类型，读者将能够掌握自定义数据类型的方法。同时，介绍了 VBA 中常量和变量的概念，能够了解变量的作用域和存活期，能够正确地对变量进行声明。编程，离不开表达式，本章介绍了 VBA 的常用表达式，相信通过学习，读者将能够在程序中灵活地运用各种表达式。

4.5 本 章 习 题

1．将下面过程运行 5 次后，在“立即窗口”中显示的最终计算结果是：（　　）

```
Public Sub test()
    Stactic C As Long
     c=c+1
     Debug.Print c
End Sub
```

A．1　　B．4　　C．5　　D．没有任何输出结果

2．VBA 常用的数据类型有哪些？

3．赋值运算符如何使用？

4．Variant 表达式和字符串表达式有何区别？

5．编写一个程序，在“立即窗口”中显示文字“姓名：王平”。

第 5 章　VBA 语句

VBA 程序由各种各样的语句构成，其中常用的语句有赋值语句和注释语句，赋值语句用于为程序中的变量保存值，注释语句用于帮助理解程序，不会产生实际的编译代码。常用的语句还有输入语句和输出语句，还可以在程序中控制使用暂停语句暂停程序的运行，使用退出语句终止程序的运行。本章的主要内容和学习目的有：

- 认识 VBA 中的语句，并学习语句的书写规则；
- 理解赋值语句的作用，学习使用赋值语句为变量赋值；
- 掌握 InputBox 函数的使用方法，能够提供用户的输入信息；
- 掌握 MsgBox 函数的使用方法，能够输出程序的结果或者返回提示信息；
- 掌握 Stop 语句和 End 语句，能够使用其暂停或终止程序的运行。

5.1　VBA 中的语句

任何一种程序设计语言都有一整套严格的编程规范。在进行代码的编写前，应该了解这些规则，使自己的代码符合这些规则，这样才能被正确地识别和执行。赋值语句和注释语句是 VBA 程序中经常用到的两类语句。程序功能的实现离不开变量和属性的赋值，为了使大型程序便于阅读，程序中也会经常出现注释语句。本节将介绍 VBA 的语句特点以及复制语句和注释语句的使用方法。

5.1.1　什么是语句

VBA 的语句是执行具体任务的指令，是 VBA 的方法、属性、函数、表达式和所有能被识别的组合。编写代码时必须遵循的规则称为语法。VBA 为了方便语句的输入，提供了语句自动格式化功能，其能够在输入 VBA 语句后，自动按照一定的规则对语句进行简单的格式化。

VBA 语句的自动格式化，包括关键字首字母自动大写、运算符前后自动输入空格以及删除语句中多余的空格等。在书写 VBA 程序代码时，必须遵循一些基本规则。遵循语句书写的基本规则，能够使程序的结构清晰、便于理解且方便调试。

一般情况下，程序中的一个语句占用一行。在 VBA 中，也可以将几个语句放在同一行中构成一个复合语句。复合语句中的各个语句使用冒号“：”来分隔。复合语句代码如下所示。

```
Debug.Print 30 : Debug.Print 31:Debug.Print 32
```

在 Visual Basic 编辑器的“代码”窗口中，每行 VBA 代码可以包含 1023 个字符，但有时语句过长，需要换行，此时可使用续行符来实现。续行符是一个空格后面加一个下划线“_”，其具体的使用方法如下所示。

```
myDocument.Shapes.Range(Array(1, 3)).Fill.Patterned _
msoPatternHorizontalBrick
```

在编写程序代码时，关键字、变量名、常量名、过程名之间一定要使用空格来进行分隔，并且应该使用缩进格式。具有缩进格式的程序有更强的可读性，能够反映程序代码的逻辑关系和嵌套关系，示例如下所示。

```
If IsNumeric(TextBox1.Text) Then                   '判断文本是否为数字
    TempNum = CInt(TextBox1.Text)                  '复合语句
    If TempNum >= 0 And TempNum <= 100 Then        '嵌套 If 语句
        ScrollBar1.SmallChange = TempNum
    Else
    TextBox1.Text = ScrollBar1.SmallChange
    End If
Else                                               '不满足条件时执行
    TextBox1.Text = ScrollBar1.SmallChange
End If
```

【**范例 5-1**】 通过示例熟悉 VBA 程序中语句的书写规范。本程序将实现对工作表中合并单元格个数的统计，同时将以提示对话框的形式显示结果，代码如下所示。

```
01  Sub 统计合并单元格()
02      Dim rng As Range, rngA As Range, i As Byte '声明变量
03      For Each rng In ActiveSheet.UsedRange      '遍历使用过的单元格
04          If rng.MergeCells Then                 '如果包含合并单元格
05              If rng.Address = Split(rng.MergeArea.Address, ":")(0) Then
                                                   '获取单元格地址后
06                  If rngA Is Nothing Then        '如果对象变量为空
07                      Set rngA = rng             '合并单元格赋予对象变量
08                  Else
09                      Set rngA = Application.Union(rngA, rng)
                                                   '合并单元格区域
10                  End If
11                  i = i + 1                      '计数变量加 1
12              End If
13          End If
14      Next
15      MsgBox "当前工作表中共有：" & i & "个合并单元格！" _
16      & Chr(10) & "合并单元格的地址为：" & Chr(10) & rngA.Address
                                                   '显示提示
17  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，程序给出提示对话框，对话框显示工作表中合并单元格的数量和地址，如图 5.1 所示。

【代码解析】本示例程序代码用来演示程序中语句的书写规范。在程序中，诸如 Dim、MsgBox 和 If 等关键字，无论输入时是大写还是小写，在输入完成后均会自动更改大小写。程序的第 15~16 行提示对话框的文字内容比较多，在“代码”窗口中可以一行输入，为了阅读方便，也可以使用“_”来对程序进行换行，以多行输入。在程序中，For Each…In Next

循环结构中使用了多重的 If 结构的嵌套。代码在输入时使用缩进格式，这样能够使程序条理清晰，读者能够很容易地理解各层嵌套之间的关系。

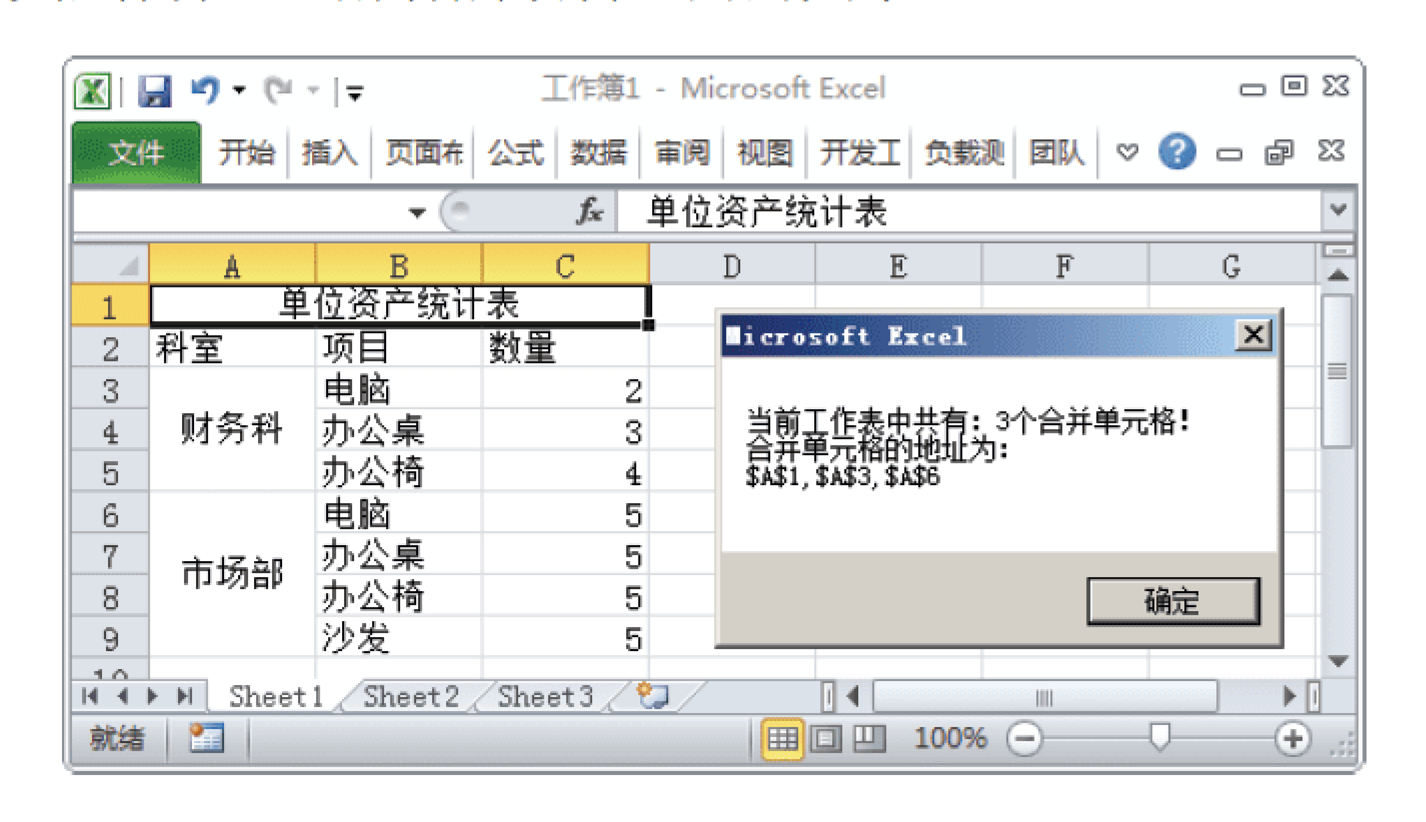


图 5.1　程序运行的效果

提示： 程序中，如果 MergeCells 属性值为 True，表示区域包含合并单元格。MergeArea 属性能够返回一个 Range 对象（即工作表对象）代表包含指定单元格的合并区域。第 05 行代码中使用 Slipt 函数来截取符合条件的单元格地址。第 09 行代码使用 Application 对象的 Union 方法来获得合并单元格区域。

5.1.2　使用赋值语句

赋值语句是 VBA 程序中最基本最常用的语句。赋值语句的作用是对表达式进行运算，同时将运算的结果赋予其左侧的变量或对象属性。赋值语句实际上在前面的各章示例中都曾经用到，它的语法格式如下：

```
[Set]<变量名>=<表达式>
```

参数说明如下所示。

- Set：可选。赋值关键字，常常省略不写。
- 变量名：必需。变量或对象属性的名称。
- 表达式：必需。赋给变量或对象属性的值。

在 VBA 中，使用赋值语句应该注意赋值运算符左边只能是变量名或对象属性，不能是常量、表达式。赋值语言执行时，先对右边的表达式进行计算，然后将结果赋给左边的变量名或对象属性。在赋值语句中，关键字 Set 一般省略。语句中的变量名必须遵循标识符命名规则。只有当等号右侧的表达式是一种与左侧变量兼容的数据类型时，赋值才会成功。例如，不能将数值变量的值赋给字符串型的变量。否则在程序编译时就会出错。

注意： 赋值语句中的“=”称为赋值符号，其用于对变量的赋值而非数学中通常理解的等于。如在 VBA 中，经常可以看到这种在数学中不可能出现的表达式“m=m+1”，表达式是将变量 m 当前值加 1 后将结果赋于变量 m。

【范例 5-2】 使用赋值语句实现修改选择单元格文字的字体、字号和颜色等。代码如

下所示。

```
Sub 设置单元格文字属性()
    Selection.Font.Name = "黑体"              '设置字体
    Selection.Font.Size = 22                  '设置字号
    Selection.Font.Bold = True                '设置为黑体
    Selection.Font.Underline = True           '设置文字带有下划线
End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。被选择单元格中文字被设定为程序指定的样式，如图 5.2 所示。

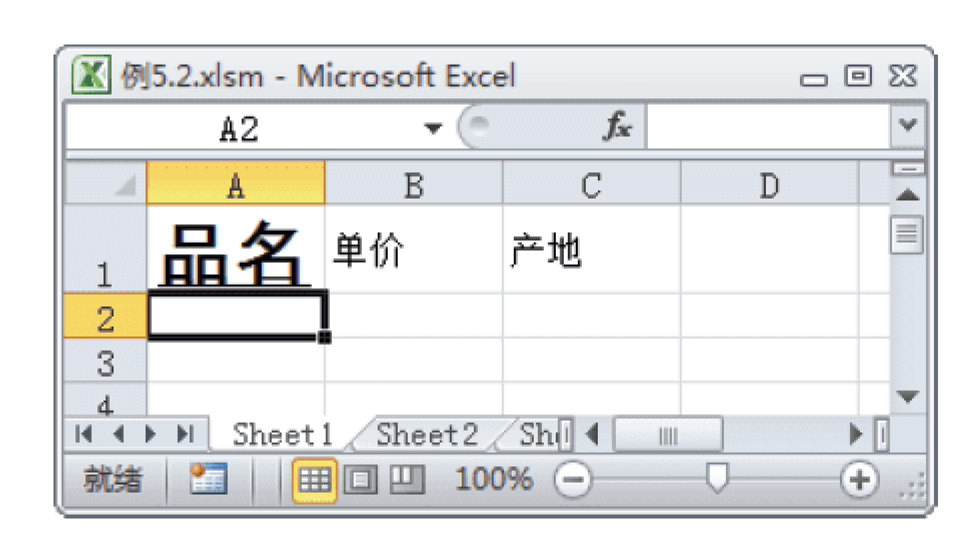


图 5.2　程序运行的效果

【代码解析】本示例程序代码用来设置单元格中文字的样式。代码中的 Name、Size、Bold 和 Underline 是 Font 对象的属性，这里使用赋值语句为这些对象属性赋值，将单元格中的文字设定为指定的样式。

注意：本段代码体现了一种最常见也是最基本的程序结构，那就是顺序结构。这种结构的程序在执行时是从上向下顺序执行的。顺序结构是比较简单的结构，比较符合人们日常生活中大多数发生的事情。

5.1.3　使用注释语句

在程序中，为了增强程序的可读性，方便程序的维护，往往需要为特定的语句添加各种说明，这种说明文字就是注释语句。注释语句对于读者来说并不陌生，在前面章节中编写有关程序代码时，已经多次使用了注释语句。注释语句一般使用下面两种格式：

```
'注释文本
```

或

```
Rem 注释文本
```

在使用单撇号（'）来添加注释语句时，只需要在注释语句前加上单撇号（'）即可，注释语句结束处不需要添加单撇号（'）。注释语句使用的示例如下面代码所示。

```
Selection.Font.ColorIndex= xlAutomatic '将文字颜色设置为自动颜色
```

使用 Rem 关键词来注释语句，应注意需要语句在程序语句和注释语句间加上冒号（:），代码如下所示：

```
Selection.Font.ColorIndex= xlAutomatic : Rem 将文字颜色设置为自动颜色
```

提示： 注释语句在程序中不会产生执行代码，其只是作为对程序的说明而存在。调试程序时，在不希望执行的代码前添加注释符号使其不被执行，这是调试程序查找语句错误的一个常用技巧。

【范例 5-3】 本实例演示注释语句的作用。其功能是在工作表中选择单元格区域，运行程序将自动选择工作表中除选择区域以外的已用单元格区域，代码如下所示。

```
Sub 反向选择单元格区域()
    Application.DisplayAlerts = False   '禁用警告提示
    Application.ScreenUpdating = False  '禁用屏幕刷新
    Dim a As String, t As String        '声明变量
    a = Selection.Address               '获得选区地址
    t = ActiveSheet.UsedRange.Address   '获得已使用单元格地址
    With Sheets.Add                     '添加一个新工作表
        .Range(t) = 0                   '新工作表中对应已用地址单元格赋值
        .Range(a) = "=0"                '新工作表中对应选区单元格赋值
        a = .Range(t).SpecialCells(xlCellTypeConstants, 1)
        .Address                        '将变量 a 设置为含有常量的单元格地址
        .Delete                         '删除新工作表
    End With
    ActiveSheet.Range(a).Select         '反向选择已用单元格区域
    Application.ScreenUpdating = True   '重新启用屏幕刷新
    Application.DisplayAlerts = True    '重新启用警告提示
End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码。切换到 Excel 2010 工作表中，选择单元格区域，如图 5.3 所示。切换回 Visual Basic 编辑器，在“代码”窗口中单击，将输入点光标放置于程序中。按 F5 键运行程序，切换到 Excel 2010 工作表中，可以看到已用单元格中未被选择的单元格区域被选择，如图 5.4 所示。

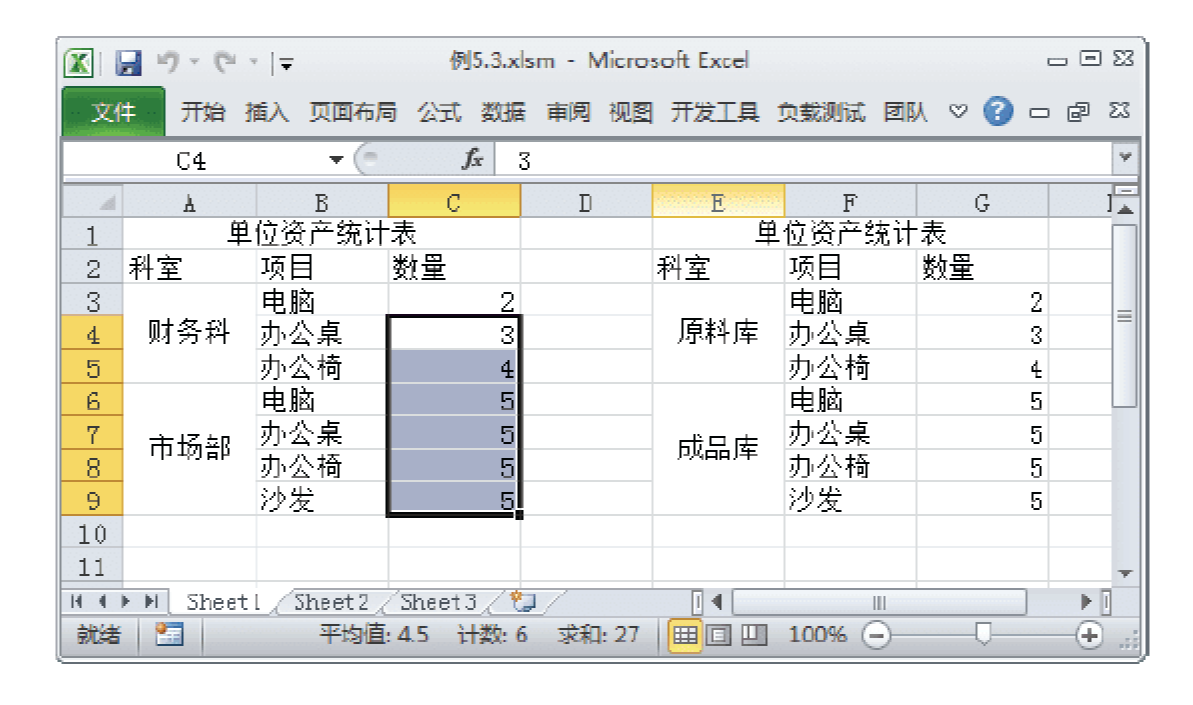




图 5.3　选择单元格区域

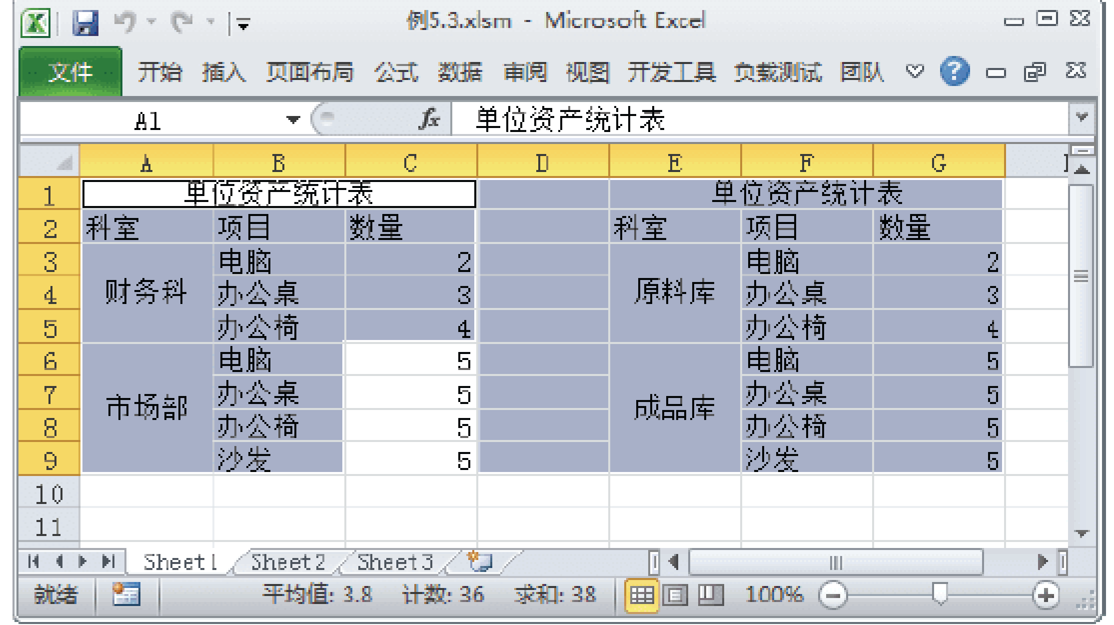


图 5.4　非选择单元格被选择

【代码解析】本示例程序代码用来演示注释语句在程序中的使用方法和意义。在这段程序中，代码较多，同时程序实现的流程较为复杂，为了能够理解代码，添加注释是一个好方法。为了使注释便于阅读，每行语句后添加注释时，可以使用 Tab 键来对齐注释。

5.2　数据的输入和输出

程序是用来处理数据的，在处理数据时，程序需要知道处理什么数据以及处理的结果如何告知用户，这就是数据的输入和输出的问题。在基于 Excel 的 VBA 程序中，数据输入和输出的方式很多，如使用工作表和用户窗体等。本节将介绍 VBA 中常见的数据输入和输出方法。

5.2.1　输入对话框

在 VBA 中，InputBox 函数能产生一个接受用户输入的对话框，用户可以在对话框中输入需要的数据，数据将传递给程序。Input 函数的语法格式如下：

```
InputBox (prompt[, title] [,default] [,Xpos] [,Ypos] [,helpfile,context])
```

参数说明如下所示。

- ❑ prompt：该参数的值是对话框中显示的提示信息，一般使用的是字符串表达式，字符串最大长度是 1024 个字符。
- ❑ title：该参数决定对话框标题栏中显示的内容，其值为字符串型数据。省略该参数，对话框标题栏将显示应用程序名。
- ❑ Default：该参数对话框中输入文本框中显示的字符串。省略该参数，输入文本框为空。
- ❑ Xpos：该参数决定了对话框的上边界距离屏幕左侧的水平距离。
- ❑ Ypos：该参数决定了对话框的上边界距离屏幕上边界的垂直距离。
- ❑ helpfile：该参数用于设置对话框的帮助信息，此参数可以省略。
- ❑ context：该参数用于设置对话框帮助主题编号，此参数可以省略。

【范例 5-4】 编程创建一个用于输入产品编号的文本框，代码如下所示。

```
01  Sub 使用 InputBox 函数()
02      Dim msg As String, title As String
03      Dim default As String, MyValue As String     '声明变量
04      msg = "请输入产品编号"                          '对话框的提示信息
05      title = "产品编号输入系统"                      '对话框标题栏文字
06      default = "A0132008803"                       '输入文本框的默认值
07      MyValue = InputBox(msg, title, default, 100, 150)  '显示输入对话框
08      Debug.Print MyValue                           '显示输入值
09  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在屏幕的左上角会显示一个输入对话框，如图 5.5 所示。在文本框中输入数据后，单击“确定”按钮关闭对话框，可以在“立即窗口”中查看输入的值，如图 5.6 所示。

【代码解析】在代码中，InputBox 函数需要的参数放置于变量中，这样便于程序的修改。在第 07 行代码中 InputBox 函数中的数字参数 100 和 150 指定了对话框显示在屏幕上

的位置。由于设置了函数的 default 参数，运行时对话框中输入框的默认值是蓝色显示，此时只需要按键即可开始输入。

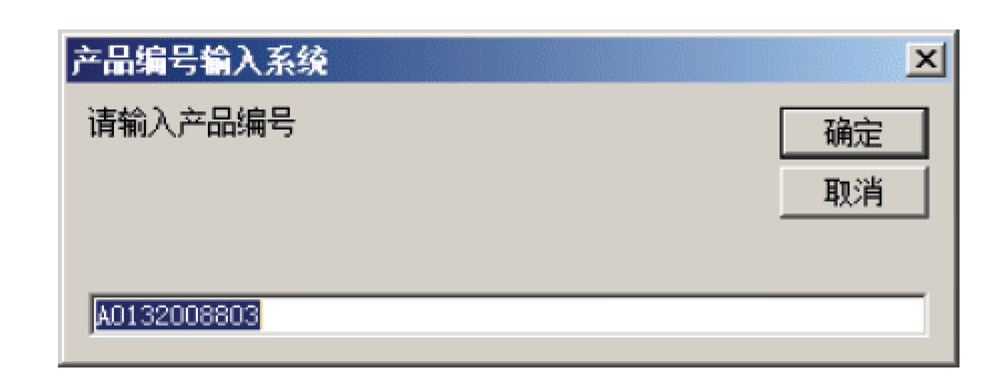


图 5.5　程序运行时获得的输入对话框



图 5.6 “立即窗口”中显示的输入值

注意：在默认情况下，InputBox 函数的返回值是字符串类型数据，如果需要使用该函数来输入数字，需要使用 Val 函数将返回值转换为相应数据类型。如果单击了对话框中的“取消”按钮，则对话框将返回一个空字符串，可以根据这个空字符串来判断用户是否有输入。InputBox 函数一次只能实现一个数据的输入，如果需要要输入多个数据，则需要多次调用该函数。

5.2.2　提示对话框

对话框是实现程序与用户交互的常用方法，使用 MsgBox 函数能产生一个对话框用于显示信息。通过单击对话框中的按钮，将能够返回数值表明用户单击的是哪个按钮，而程序也将继续执行。MsgBox 函数使用的语法格式如下：

```
value=MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

MsgBox 函数的参数和 InputBox 函数的参数的用法相同，这里不再赘述。这里要说明的是，如果不需要通过返回值获得用户单击的按钮，可以直接使用函数而不要上述语句中的对参数赋值部分。

在 MsgBox 函数中增加了一个 buttons 参数，该参数用于指定按钮显示的数目、样式和消息对话框显示的图标样式等。要了解有哪些常量可以使用，可以在 VBA 自带的帮助文档查询 MsgBox 的提示信息。在帮助文档中查询到的部分 buttons 参数的意义，如图 5.7 所示。

【范例 5-5】 使用消息对话框显示单元格中输入的内容，对话框包含“重试”按钮和“取消”按钮，对话框使用警告样式，代码如下所示。

```
Sub 使用MsgBox函数()
    Dim x As String                                  '声明变量
    x = Selection.Value                              '获取单元格中的文字
    y = MsgBox("你选择的单元格中的文字是" & "“" & _
    x & "”", vbRetryCancel + vbExclamation, "提示") '显示提示信息
    Debug.Print y                                      '显示单击的按钮的值
 End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在屏幕的左上角会显示一个对话框。对话框包含“重试”和“取消”按钮，同时将显示选择单元格中文字内容，如图 5.8 所示。单击“重试”按钮关闭对话框，在“立即窗口”中

可以查看“重试”按钮对应的返回值，如图 5.9 所示。

常数	值	描述
vbOKOnly	0	只显示 OK 按钮。
VbOKCancel	1	显示 OK 及 Cancel 按钮。
VbAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮。
VbYesNoCancel	3	显示 Yes、No 及 Cancel 按钮。
VbYesNo	4	显示 Yes 及 No 按钮。
VbRetryCancel	5	显示 Retry 及 Cancel 按钮。
VbCritical	16	显示 Critical Message 图标。
VbQuestion	32	显示 Warning Query 图标。
VbExclamation	48	显示 Warning Message 图标。
VbInformation	64	显示 Information Message 图标。
vbDefaultButton1	0	第一个按钮是缺省值。
vbDefaultButton2	256	第二个按钮是缺省值。
vbDefaultButton3	512	第三个按钮是缺省值。
vbDefaultButton4	768	第四个按钮是缺省值。
vbApplicationModal	0	应用程序强制返回；应用程序一直被挂起，直到用户对消息框作出响应才继续工作。
vbSystemModal	4096	系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工作。
vbMsgBoxHelpButton	16384	将Help按钮添加到消息框

图 5.7　帮助信息中的 buttons 参数列表

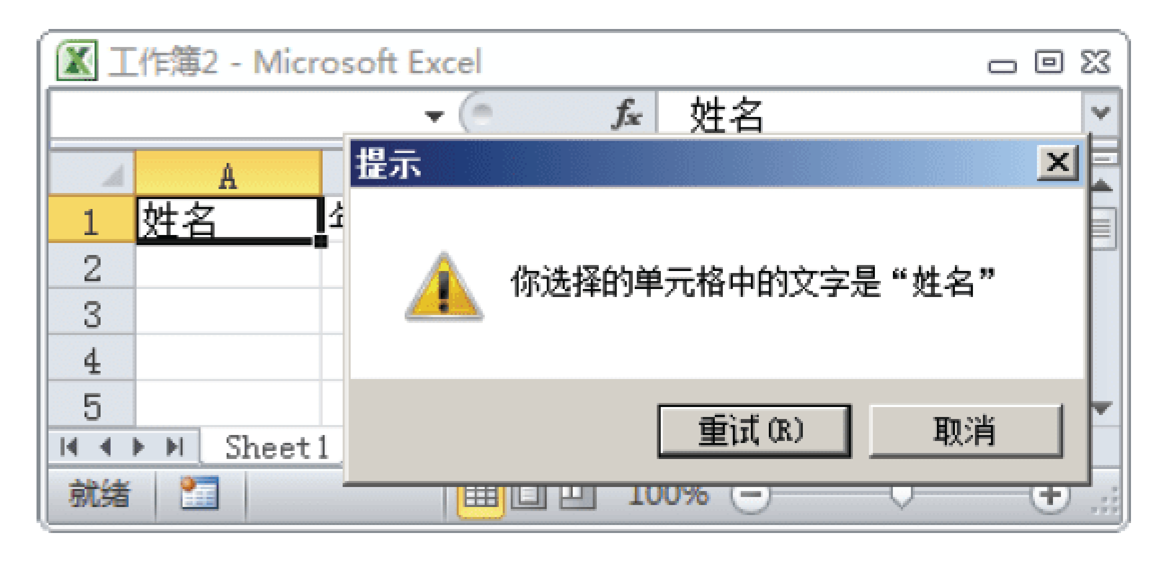



图 5.8　程序运行时获得的提示对话框



图 5.9　“立即窗口”中显示的结果

【代码解析】本段代码用于演示 MsgBox 函数的使用特点。为了使单元格中文字在对话框中显示时能够放置于引号中，在设置 MsgBox 的显示信息文字时，使用&运算符来连接左引号““”、单元格中文字内容（即变量 x 值）和右引号“””。在设置 MsgBox 函数的 button 参数值时，如果需要使用多个按钮，可以像示例中那样用“+”连接多个常量来进行设置。单击对话框中的不同按钮对应的返回值如图 5.10 所示。这个返回值同样可以在 VBA 自带的帮助文档中查到。

返回值

常数	值	描述
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

图 5.10　按钮对应的返回值

5.2.3　显示程序运行结果

Print 方法可用于在程序调试时显示程序运行结果，这个结果在 Visual Basic 编辑器的“立即窗口”中显示。在 VBA 中，Print 方法的语法结构如下所示：

```
Debug.Print 显示内容
```

【范例 5-6】 编程在“立即窗口”中显示 3 个间隔的 o(∩_∩)o，代码如下所示。

```
01  Sub Print 方法的使用()
02      Debug.Print "o"; "("; "∩"; "_"; "∩"; ")"; "o", "o"; "("; _
03       "∩"; " "; "∩"; ")"; "o", "o"; "("; "∩"; " "; "∩"; ")"; "o"
                                                '显示第一行图案
04      Debug.Print Spc(8); "o"; "("; "∩"; "_"; "∩"; ")"; "o"; _
05      Spc(3); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                '显示第二行图案
06      Debug.Print Spc(14); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                '显示第三行图案
07      Debug.Print Spc(8); "o"; "("; "∩"; "_"; "∩"; ")"; "o"; _
08      Spc(3); "o"; "("; "∩"; "_"; "∩"; ")"; "o"
                                                '显示第四行图案
09      Debug.Print "o"; "("; "∩"; "_"; "∩"; ")"; "o", _
10      "o"; "("; "∩"; "_"; "∩"; ")"; "o", "o"; "("; "∩"; "_"; "∩"; ")";
"o"                                             '显示第五行图案
11  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序。在“立即窗口”中可以看到程序运行的结果，如图 5.11 所示。

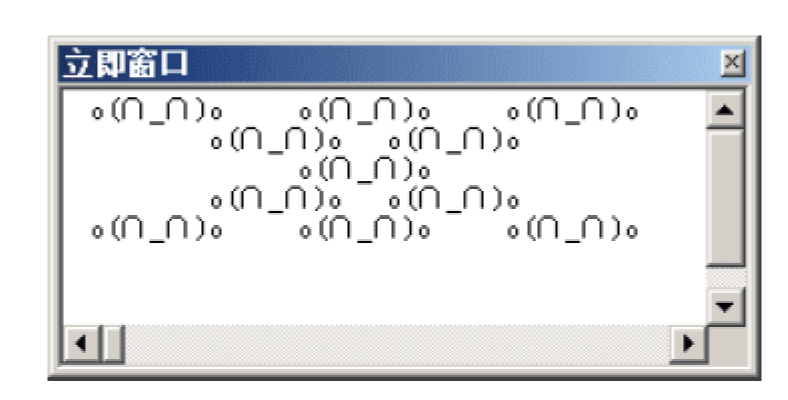


图 5.11　程序运行的效果

【代码解析】在前面章节的很多范例中都使用了 Print 方法在“立即窗口”中显示运行结果，本段代码最大的亮点在于使用了分隔符来格式化输出的数据。本例中使用了两个分隔符，一个是分号（;），其用于将前后两个数据连接在一起输出。“立即窗口”中显示的每一个 o(∩_∩)o 都是用分号(;)连接各个字符得到的。第二个使用的分隔符就是逗号(,)，其以 14 个字符为一个输入区，将每个数据输出到对应的输入区中。代码中就是使用逗号(,)创建了 3 个输入区，得到了规则排列的输出效果。

显示的第二行图案，只需要打印 2 个 o(∩_∩)o。在第 04~05 行代码中，使用 Spc（8）在这一行的前面添加 3 个空格后再拼接 2 个 o(∩_∩)o，并使它们的间隔为 3 个空格。同理，根据 o(∩_∩)o 摆放的位置，分别添加空格来使它们对齐，最终产生图 5.11 的显示效果。

提示： 对于 Print 方法来说，除了上面介绍的 2 个分隔符外，其还包括下面 2 个分隔符：一个是 Spc(n)，将其放置于 2 个需显示的字符之间，可以在这 2 个字符间插入 n 个空格；另一个是 Tab(n)，该分隔符能够移动光标，就像按 Tab 键一样，光标移动的距离由 n 决定。

5.3　程序的中断

程序开始运行后，有时需要查看运行到某个语句时的运算中间值，这时就需要暂停程序。而当程序结果满足某个条件时，有时又需要结束程序的运行。要在程序中实现上面提到的两个功能，需要使用暂停语句和退出语句。

5.3.1　暂停程序

当需要程序在执行到某个语句暂停时，可以在该语句处放置一个 Stop 语句使程序的运行暂停。这里要注意的是，Stop 语句是暂停程序的运行而不是退出程序，因此其不会关闭程序，变量也不会被清除，在需要时可以从暂停处开始继续程序的运行。

Stop 语句相当于在程序中设置了一个断点，因此其常用在程序的调试过程中，下面举一个示例来了解 Stop 语句在程序中的作用。

【范例 5-7】 使用 Stop 语句来实现循环的暂停，代码如下所示。

```
Sub 使用暂停语句()
    Dim n As Integer                                      '声明变量
    For n = 1 To 10
        n = n + 1                                         '变量加 1
        Debug.Print n                                     '显示变量值
        Stop                                              '暂停程序
    Next
End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，程序运行到 Stop 语句处会暂停，在“代码”窗口中会标示出程序暂停的位置。同时，在“立即窗口”中可以看到当前 n 的值。再次按 F5 键程序将继续运行，在“代码”窗口中再次显示新的 n 的值，如图 5.12 所示。

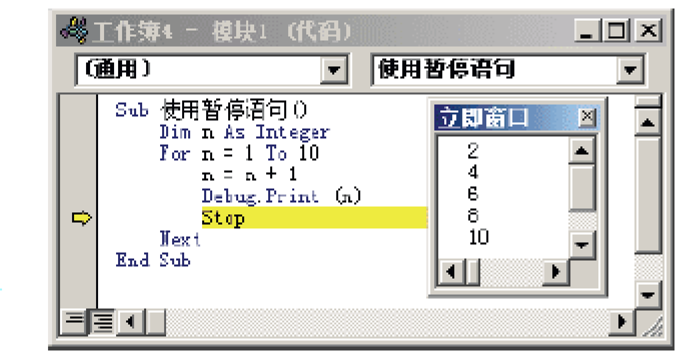


图 5.12　程序运行的效果

【代码解析】本段程序将显示 Stop 语句所起的作用。在程序运行时，运行到循环体中的 Stop 语句时，程序将暂停，暂停前 Print 语句在“立即窗口”中显示 n 的当前值。此时，变量 n 的值并没有清除，当程序结束暂停再次运行时，变量的值将在原有值的基础上增加 2。

5.3.2　停止程序

在 VBA 中，可以使用 End 语句来停止程序的运行。此时，程序的运行将被终止，返回到 VBA 编辑器。此时，VBA 程序将卸载所有窗体，在没有其他程序引用当前程序的公共类模块所创建的对象且无代码执行的情况下，变量内存将被清空，程序将立即关闭。下面对范例 5-7 的程序代码进行修改，使用 End 语句来停止程序的运行，比较 End 语句与 Stop 语句的不同。

【范例 5-8】 使用 End 语句来实现循环的暂停，代码如下所示。

```
01  Sub 退出程序()
02      Dim n As Integer                    '声明变量
03      For n = 1 To 10
04          n = n + 1                       '变量加 1
05          Debug.Print n                   '显示变量值
06          End                             '退出程序
07      Next
08  End Sub
```

【运行结果】创建一个模块，在模块的“代码”窗口输入上述代码，按 F5 键运行程序，在“立即窗口”中可以看到当前 n 的值。再次按 F5 键程序将重新启动，此时在“代码”窗口中显示的 n 的值仍然是 2，如图 5.13 所示。

【代码解析】在程序运行时，变量 n 的初始值为 0。在运行第 05 行的 End 语句前的语句后“立即窗口”中显示 n 的值为 2。执行 End 语句时，程序将退出，变量将被清除。当再次运行这段程序时，程序中变量的初始值仍然为 0，与上一次程序运行一样执行的是相同的语句，因此得到的结果与第一次执行的结果相同。

图 5.13　程序运行的效果

提示：实际上在 VBA 中，使用 Quit 方法同样能够退出程序，但此时将退出 Excel 系统而不只是回到 VBA 编辑器。在使用此方法时，如果处于打开状态的工作簿还没有保存，Excel 会给出提示对话框提示保存文档。

5.4　小　　结

本章介绍了 VBA 编程的基础知识，包括 VBA 的语法规则、赋值语句和注释语句的使用要点、数据的输入和输出的方法以及在代码中结束程序运行的方法。通过本章的学习，读者将能熟练地对变量进行赋值，能够使用 InputBox 函数实现用户数据的输入，能够采用不同的方法输出程序处理结果，同时能够在程序调试中灵活使用 Stop 语句来暂停程序。

到目前为止，读者已经能够使用 VBA 来解决很多问题，特别是数值计算的问题。但仅仅掌握这些是不够的，还需要掌握各种程序结构及其实现方式，才能编写功能更为强大的应用程序。第 6 章中，我们将一起学习 VBA 程序的常见控制结构，在 VBA 编程技术上更上层楼。

5.5　本 章 习 题

1．下列语句哪个在程序中会被视为注释？（　　）

A．声明变量　　B．'start A　　C．--sleep　　D．“join me”

2．下面哪个语句能够显示一个标题栏为“我的输入框”的输入对话框？（　　）

A．a=InputBox("请在此输入数值","我的输入对话框",30)

B．a=InputBox("我的输入对话框",)

C．a=InputBox("我的输入对话框",30)

D．a=InputBox("我的输入对话框",30,30)

3．下面哪个语句能够获得一个提示对话框，对话框的标题为“重要提示”，提示内容是“输入错误！”，对话框包含“是”和“否”按钮。（　　）

A．a=MsgBox("重要提示","输入错误！",vbYesNo)

B．a=MsgBox("重要提示","输入错误！",vbOKCancel)

C．a=MsgBox("输入错误！",vbYesNo, "重要提示")

D．a=MsgBox("输入错误！",vbOKCancle, "重要提示")

4．下面哪个语句能够在“立即窗口”中显示同一行中 3 个等距排列字符“OK”。（　　）

A．Debug.Print "OK"_"OK"_"OK"　　B．Debug.Print "OK";"OK";"OK"

C．Debug.Print OK,OK,OK　　D．Debug.Print "OK","OK","OK"

5．编写一个简单的乘方计算器，用户输入底数和指数，运算结果在提示对话框中显示。

【提示】代码中需要注意的是 InputBox 函数返回的是字符串数据，而运算符^对数值型数据进行计算。因此这里使用 Val 函数来进行数据类型的转换。

6．编写一个程序，使程序在“立即窗口”中绘制数字三角形，绘制三角形使用的数字由用户设置。

【提示】此练习主要巩固 InputBox 函数的使用方法和 Print 方法的分隔符的使用技巧。

第 6 章　VBA 程序控制结构

程序是按照一定的顺序来执行的。程序在执行过程中，可以按顺序执行，可以有选择地执行，也可以按重复执行某一段代码，还可以跳过某些代码段而执行其他代码。在 VBA 中还提供了 With 结构语句用于简化代码的编辑，同时在程序执行过程中还可以捕获程序执行的异常。本章的主要学习内容和学习目的如下：

- 学习 VBA 程序的控制，掌握程序的走向；
- 学习循环控制，使用循环结构编写程序；
- 学习 With、Exit 和 GoTo 等语句；
- 学会调试程序，掌握对 VBA 程序的异常处理技术。

6.1　使用 VBA 选择结构

在实际的程序编写过程中，往往需要对条件进行判断，根据判断的结果来控制程序的流程，此时就需要使用选择结构控制程序执行。选择结构是程序的一种常见结构，本节将对 VBA 中的选择结构语句进行介绍。

6.1.1　程序的结构

在使用 VBA 开发应用程序过程中，结构化设计是最为基本的程序设计方法，而程序控制结构是结构化设计的基础。合理地使用控制结构，可以使程序结构清晰，增强程序的“健壮性”。VBA 提供了 3 种基本的程序控制结构，它们是顺序结构、选择结构以及循环结构。

在 VBA 中，顺序结构是按照语句书写的顺序从上到下逐条执行的程序结构。在程序执行过程中，排在前面的语句先执行，排在后面的语句后执行，执行过程中没有任何分支。顺序结构是最为常见的程序结构，也是其他两种结构的基础。前面章节中大多数的示例都是顺序结构的程序。顺序结构的执行流程如图 6.1 所示。

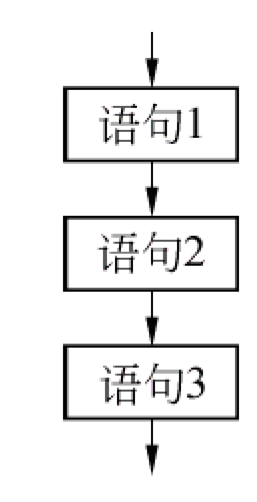


图 6.1　顺序结构流程

选择结构也称为分支结构。此类程序结构具有分支，由条件来决定应该执行哪个分支中的语句。选择结构可以是二分支或多分支结构，分支还可以进行嵌套。典型的多分支结构程序的执行流程如图 6.2 所示。

循环结构的程序在运行时将重复执行某一部分代码以完成大量有规律的重复运算。循环结构的程序的分支比选择结构要复杂，其典型的执行流程如图 6.3 所示。

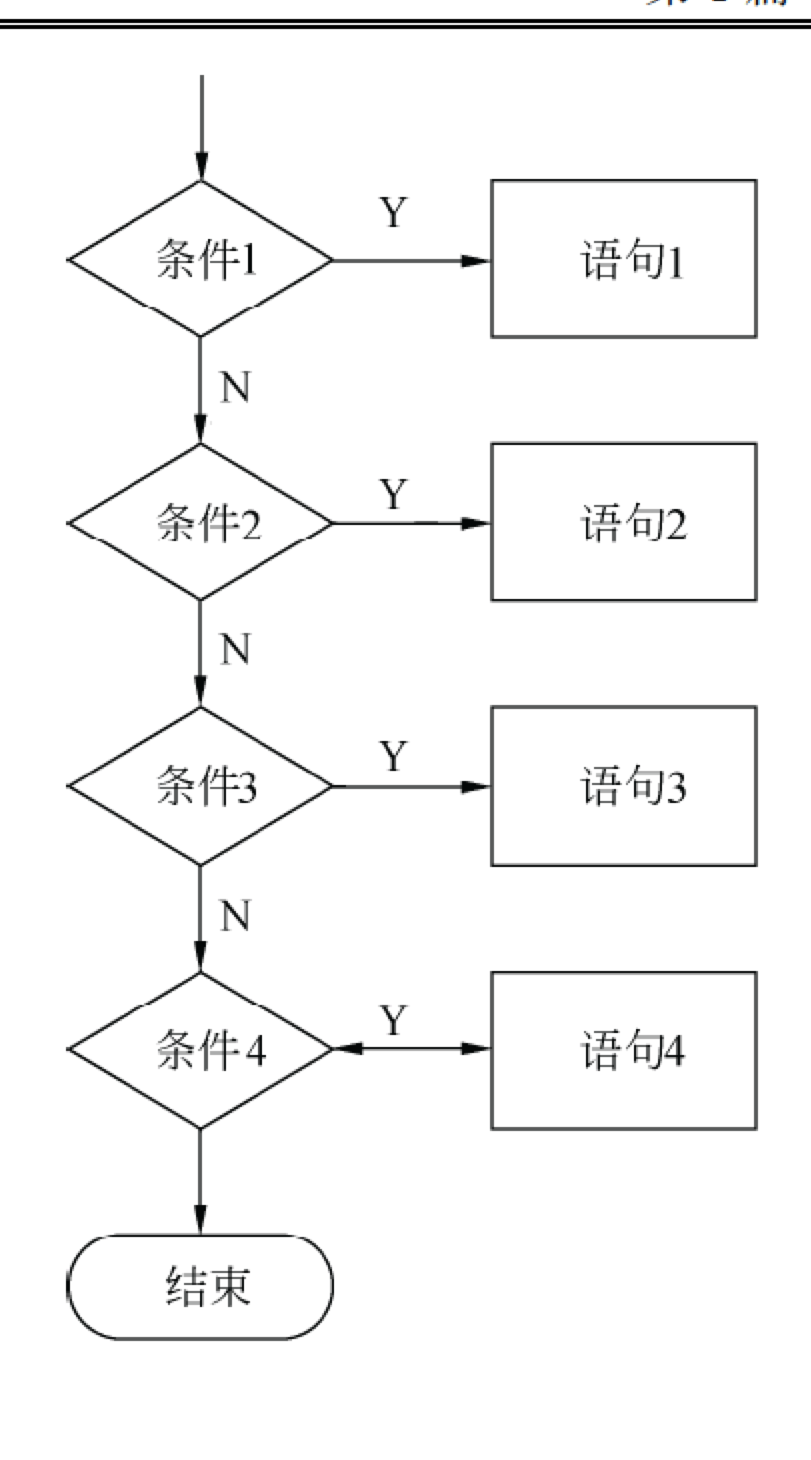


图 6.2　多分支结构的流程线

图 6.3　循环结构的流程线结构

提示： VBA 语句的自动格式化包括关键字首字母自动大写、运算符前后自动输入空格以及删除语句中多余的空格等。如在输入关键字 Dim 时，无论输入 dim，还是 DIM 或是 dIm，当输入完成换行后，Visual Basic 编辑器都会自动将其变为符合规范的 Dim。

6.1.2　使用条件表达式

在使用选择结构时，需要使用条件表达式描述条件。条件表达式是使用逻辑运算符或比较运算符连接的表达式，根据所使用运算符的不同，表达式可分为关系表达式和逻辑表达式。对于条件表达式来说，其运算结果只有两个，即真（True）和假（False）。其具体示例如下所示。

```
Len (a) > 20 Or Len(a) < 10
```

此表达式一个关系表达式，如果变量 a 字符串的字符数目大于 20 或者字符串的字符数目小于 10 时，其结果为真（True）。

```
Not(1<>5)
```

此表达式先计算 1<>5，其值为真（True），然后使用逻辑运算符 Not 对结果取反，表达式的最终结果为假（False）。

```
Range("A1") ="武汉" And Range("B1") = "男"
```

此表达式只有当前后两个关系表达式的值都是真（True）时，其值才为真（True）。

6.1.3　If…Then 语句

程序在运行时往往需要针对不同的结果执行不同的操作。例如，根据天气决定是否出行，如果是晴天，则去旅行。If...Then 语句能够执行这样的判断，其允许在程序中判断一个条件，然后根据此条件的结果运行不同的代码段。

在 VBA 的程序中，If…Then 语句可以判断表达式的值，当表达式的值满足条件时执行其中包含的一组程序。If…Then 语句分为单行结构条件语句和块结构条件语句这两类。单行结构条件语句是最基本的条件语句，其语法格式为：

```
If 表达式 Then 语句
```

参数说明如下所示。

- 表达式：必需。表达式运算结果是进行条件判断的依据。如果表达式的结果为 True，则执行语句组；表达式的结果为 False，则不执行。
- 语句：可选，由一条或多条语句组成。当使用多条语句时，语句间应该使用冒号（:）连接。

在 If…Then 语句中，如果条件成立时需要执行多条语句，则可使用区块结构的方式来书写多条语句。区块结构的条件语句的语法格式如下：

```
If 表达式 Then
   语句组
End If
```

使用 If…Then 语句时，当作为条件的表达式值为真时，Then 后的语句将被执行。否则，这些语句将不会被执行，而是执行 If…Then 语句下面的程序。If…Then 语句的执行流程如图 6.4 所示。

注意：块结构的条件语句中的参数与单行结构的条件语句相同，其功能也是一样的。但要注意块结构的条件语句在结束处必须要添加“End If”语句，否则程序会提示语法错误。

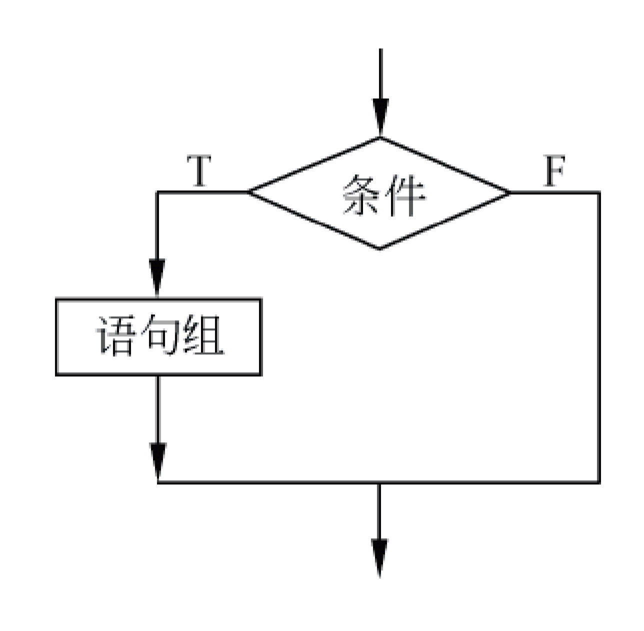


图 6.4　If…Then 语句的流程图

【范例 6-1】 编写程序，对输入的产品价格进行限制，如果价格过低就给出提示，代码如下所示。

```
Sub 条件语句示例 1()
    Dim a As Integer                                    '声明变量
    a = InputBox("请输入产品价格：")                    '输入价格
    If a < 1.2 Then                                     '比较输入值
        MsgBox "输入价格过低，此价格无法录入系统！"     '给出提示
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序。程序给出输入对话框要求输入数据，在对话框中输入产品价格，如图 6.5 所示。单击“确

定”按钮关闭对话框，如果输入的数据小于程序中指定的 1.2，程序给出提示，如图 6.6 所示。

【代码解析】本段示例代码用于演示 If…Then 语句的使用方法。在该段代码中使用 InputBox 函数，由用户输入数据，使用 If 语句来判断数据与设定数值的大小关系，此处是 1.2。如果小于 1.2，则使用 MsgBox 函数给出提示。



图 6.5　在输入框中输入数据



图 6.6　输入数据偏小则显示提示

6.1.4　If…Then…Else 语句

在编程中，往往希望程序提供判断能力，在符合某个条件（即当条件为 True 时），就执行某些代码，如果不符合某个条件，则执行其他代码。在使用 If…Then 语句时，如果条件值为假时，将退出选择结构，继续执行其下的语句。如果需要在条件为假时不退出选择结构而是执行某些程序语句，则应该使用 If…Then…Else 语句。If…Then…Else 语句的语法格式如下：

```
If 表达式 Then
    语句组 1
Else
    语句组 2
End If
```

参数说明如下所示。

- 表达式：必需。表达式运算结果是进行条件判断的依据。如果表达式的结果为 True，则执行语句组 1；表达式的结果为 False，则执行语句组 2。
- 语句组 1：可选，由一条或多条语句组成，表达式的值为 True 时执行这个语句组。
- 语句组 2：可选，由一条或多条语句组成，表达式的值为 False 时执行这个语句组。

If…Then…Else 语句还具有一种单行格式，这种单行格式适用于语句组中语句都是单行语句的情况，其语法格式如下所示。

```
If<表达式>Then[语句组 1]Else[语句组 2]
```

程序运行时，VBA 首先判断作为条件的表达式的值，如果其为 True 则执行“语句组 1”中的语句。如果其值为 False，则执行 Else 语句块（即“语句组 2”）中的语句，其执行流程如图 6.7 所示。

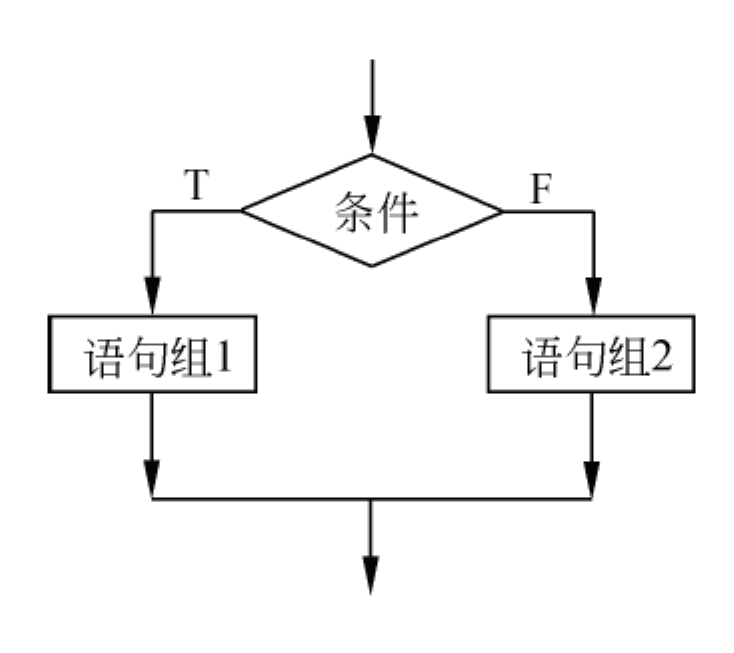


图 6.7　If...Then…Else 语句流程

【范例 6-2】 对成绩表单元格 B2 中的成绩进行判断，大于等于 60 分，评定等级为及格，否则评定等级为不及格，将评定的等级结果写入 C2 单元格，代码如下所示。

```
Sub 判断成绩是否及格()
    If Range("b2") > =60 Then                    '判断单元格中的分数是否大于 60
        Range("c2").Value = "及格"               '大于或等于 60，单元格显示及格
    Else
        Range("c2").Value = "不及格"             '不大于等于 60，单元格显示不及格
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，在工作表的 C1 单元格中显示判断的结果，如图 6.8 所示。

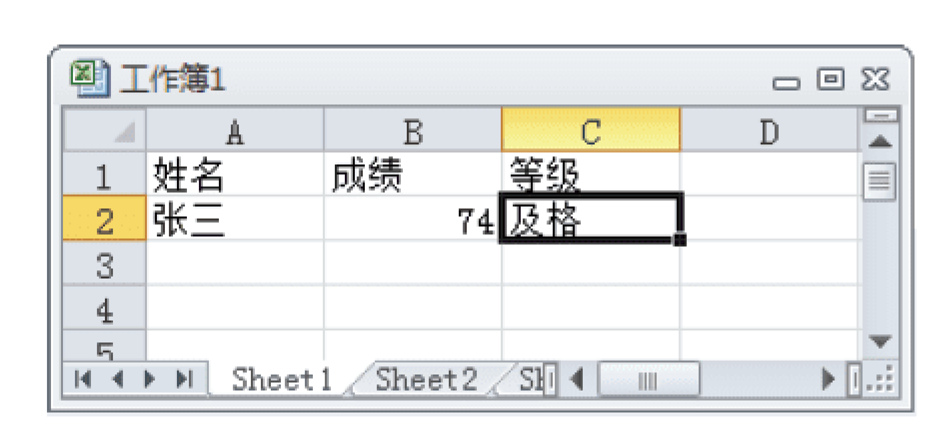


图 6.8　程序运行结果

【代码解析】这段代码用于演示 If…Then…Else 语句的使用方法。在该段代码中，以 B2 单元格的值是否大于或等于 60 作为判断条件。当其值为真时，C2 单元格设置为“及格”，否则为“不及格”。

提示：在使用条件语句时应注意代码书写的缩进格式。在代码中使用缩进格式将增强代码的可读性，有利于理解复杂条件关系的逻辑结构。

6.1.5　IIf 函数

IIf 函数可以用来执行简单的条件判断操作，它是“If…Then…Else”结构的简写版本。IIf 函数的作用是根据表达式的值返回所给出两个值中的一个。与 If…Then…Else 结构相比，在需要使用大量简单条件判断的程序中，使用该函数能够在完成条件判断的同时，减少程序的代码量，起到优化程序的作用。该函数的语法格式如下：

```
Result=IIf(表达式,True 部分, False 部分)
```

参数说明如下所示。

- Result：函数返回值。
- 表达式：判断条件。
- True 部分：当表达式的值为 True 时，返回这部分的值或表达式。
- False 部分：当表达式值为 False 时，返回这部分的值或表达式。

例如，在下面示例的代码中，根据变量 a 的值与 3 的大小关系赋予 a 不同的值。

```
If a>3 Then                                '判断 a 是否大于 3
    a=3                                    '若 a 大于 3 则将 a 置为 3
Else
    a=5                                    '否则将 a 值置为 5
End If
```

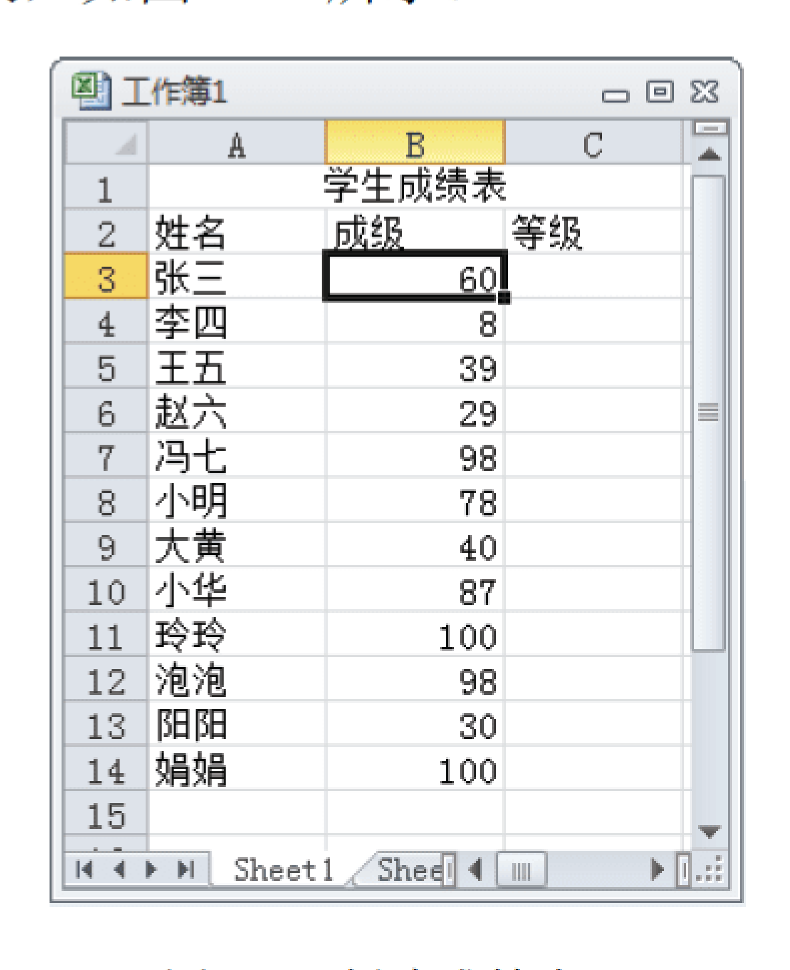
上面这段代码的功能可以简单地用 IIf 函数来完成，代码如下所示。

```
a=IIf(a>3,3,5)
```

【范例 6-3】 对成绩表“成绩”列中的成绩进行判断，看其是否大于 60 分，大于 60 分评定等级为及格，否则评定等级为不及格。将评定等级的结果写入对应的第 3 列中，代码如下所示。

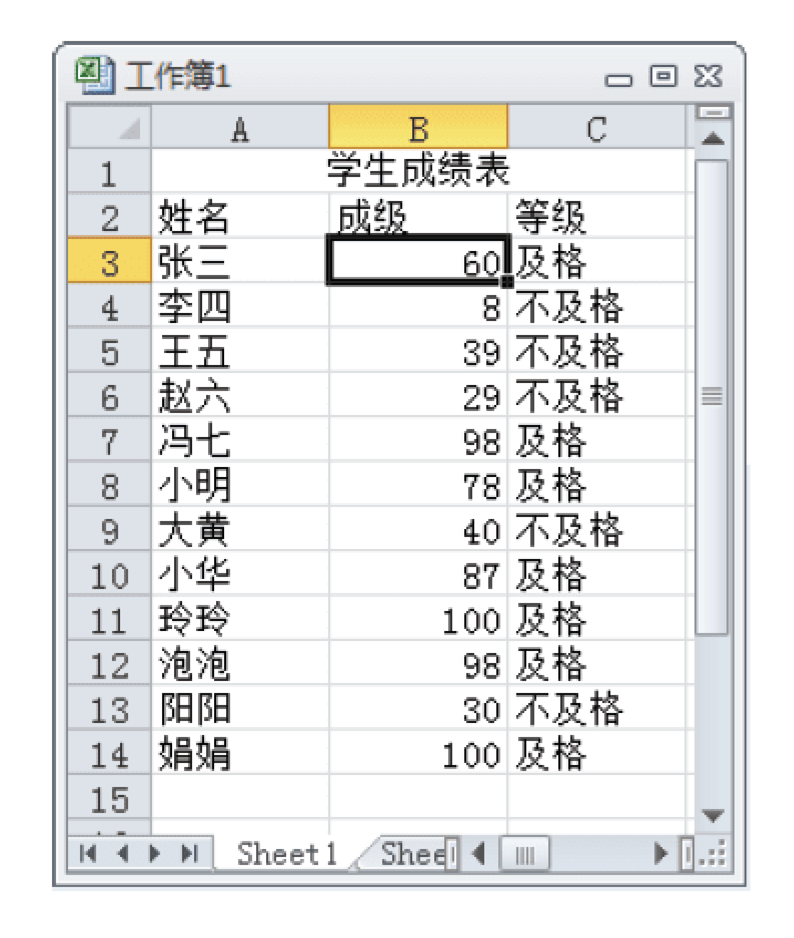
```
01  Sub iif语句的应用()
02      Dim n As Integer, rng As Range '声明变量
03      For n = 3 To Range("b1048576").End(xlUp).Row
                                        '从第 3 行开始遍历 b 列的所有非空单元格
04          Set rng = Cells(n, 2)       '指定对象变量
05          a = IIf(rng >= 60, "及格", "不及格")'判断单元格中分数等级
06          Cells(n, 2).Offset(0, 1) = a        '将等级写入“等级”单元格中
04      Next
08  End Sub
```

【运行结果】创建成绩表，如图 6.9 所示。切换到 Visual Basic 编辑器，插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，程序在“等级”列中写入学生成绩等级，如图 6.10 所示。



	A	B	C
1		学生成绩表	
2	姓名	成级	等级
3	张三	60	
4	李四	8	
5	王五	39	
6	赵六	29	
7	冯七	98	
8	小明	78	
9	大黄	40	
10	小华	87	
11	玲玲	100	
12	泡泡	98	
13	阳阳	30	
14	娟娟	100	
15			

图 6.9　创建成绩表



	A	B	C
1		学生成绩表	
2	姓名	成级	等级
3	张三	60	及格
4	李四	8	不及格
5	王五	39	不及格
6	赵六	29	不及格
7	冯七	98	及格
8	小明	78	及格
9	大黄	40	不及格
10	小华	87	及格
11	玲玲	100	及格
12	泡泡	98	及格
13	阳阳	30	不及格
14	娟娟	100	及格
15			

图 6.10　程序运行后的效果

【代码解析】这段代码用于演示 IIf 语句的使用方法。在代码的第 05 行使用 IIf 函数来实现对分数等级的评定，这里以第 2 列单元格的值是否大于或等于 60 作为判断条件。当大于等于 60 时，变量 a 的值为“及格”，否则为“不及格”。第 03~10 行代码使用 For..Next 循环结构来实现对第 2 列中从第 3 行到最后一个非空单元格所在行间的所有单元格的遍历，使用 IIf 函数依次判断这些单元格的值是否大于等于 60 分，将判断结果写入向右偏移一个列的单元格中，实现了对等级结果的写入。

6.1.6　If…Then…ElseIf 语句

在编写程序时，有时需要对多个不同的条件进行判断，根据这些条件的不同值来执行不同的代码。此时，如果逐条地使用 If...Then 语句，程序必将显得臃肿。在 VBA 中，可

以使用 If…Then…ElseIf 语句来对多个不同的条件进行判断，根据判断得到的不同结果来在多个程序块中选择执行其中的一个。If…Then…ElseIf 语句的语法格式如下：

```
If 表达式 1 Then
    语句组 1
ElseIf 表达式 2 Then
    语句组 2
.........
ElseIf 表达式 n-1Then
    语句组 n-1
Else
    语句组 n
End If
```

参数说明如下所示。

- 表达式 1：必需。表达式 1 运算结果是进行条件判断的依据。如果表达式的结果为 True，则执行语句组 1；表达式的结果为 False，则判断表达式 2。
- 语句组 1：可选，由一条或多条语句组成，表达式 1 的值为 True 时执行这个语句组。
- 表达式 2：必需。表达式 2 运算结果是进行条件判断的依据。如果表达式 2 的结果为 True，则执行语句组 2；表达式的结果为 False，则判断下一表达式。
- 语句组 2：可选，由一条或多条语句组成，表达式 2 的值为 True 时执行这个语句组。
- 语句组 n：可选，由一条或多条语句组成，表达式 1 到表达式 n-1 的值均为 False 时执行这个语句组。

注意：在 If…Then…ElseIf 语句中可以包含任意数量的 ElseIf 子句和条件，但是 ElseIf 子句必须出现在 Else 子句之前。

在上述语法结构中，VBA 首先判断“表达式 1”的值是否为 True，如果为 True 则执行“语句组 1”中的语句。如果为 False，则再判断“表达式 2”的值是否为 True，依此类推。在执行过程中，只要找到了一个为 True 的条件，就执行相应的语句块，执行完成后将执行 End If 语句后的程序。If…Then…ElseIf 语句的执行流程如图 6.11 所示。

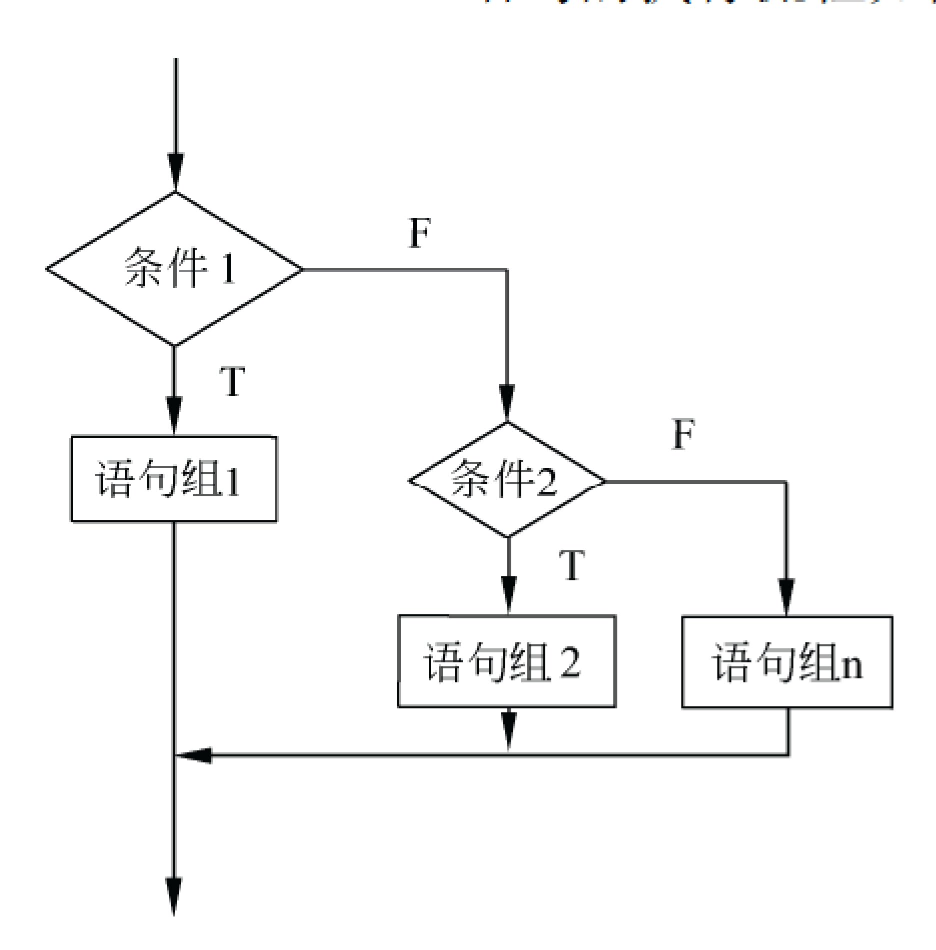


图 6.11　If…Then…ElseIf 结构的流程线

【范例 6-4】 使用 If...Then...ElseIf 语句来评定分数等级。分数小于 60 分为不及格，分数大于等于 60 小于 70 为及格，分数大于等于 70 小于 80 为中等，分数大于等于 80 小于 90 为良好，分数大于等于 90 的为优秀，代码如下所示。

```
Sub 评定分数等级()
    If Range("b2") >= 90 Then                    '判断分数是否大于等于 90
        Range("c2").Value = "优秀"               '大于等于 90，等级为优秀
    ElseIf Range("b2") >= 80 And Range("b2") < 90 Then
                                                 '判断分数是否在 80 至 90 之间
        Range("c2").Value = "良好"               '判断等级为良好
    ElseIf Range("b2") >= 70 And Range("b2") < 80 Then
                                                 '判断分数是否在 70 至 80 之间
        Range("c2").Value = "中等"               '判断等级为中等
    ElseIf Range("B2") >= 60 And Range("b2") < 70 Then
                                                 '判断分数是否在 60 至 70 之间
        Range("c2").Value = "及格"               '判断等级为及格
    Else
        Range("c2").Value = "不及格"             '条件均不满足，等级为不及格
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，工作表中显示评定的分数等级，如图 6.12 所示。

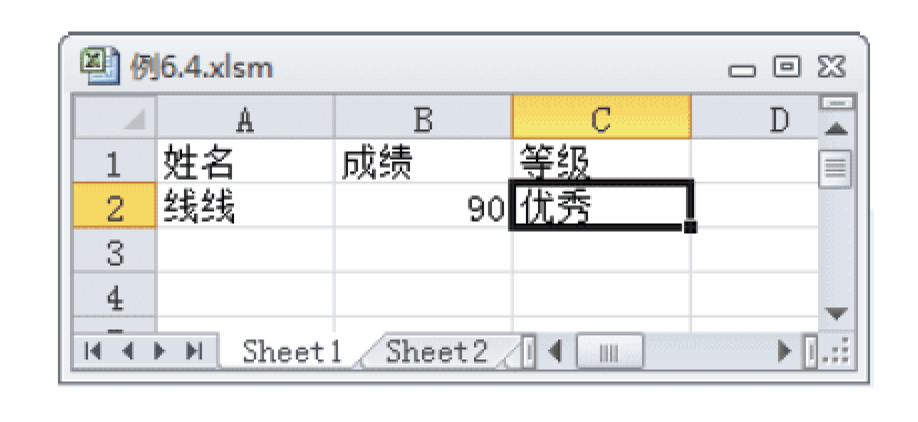


图 6.12　程序运行结果

【代码解析】示例代码使用 If...Then...ElseIf 语句来评定分数等级。作为判断条件的分数，根据等级被分为了不同的分数段，分数段的下限值取等号而上限不取等号，避免了判断条件的重复。根据判断结果的不同，将不同的等级写入相应的单元格中。

注意：这里要注意条件的书写顺序，如果以大于等于作为条件来进行判断，应该按照从高到低的顺序来进行判断。如果以小于等于来进行判断，则判断顺序正好相反。

6.1.7　Select Case 语句

Select Case 语句是实现多条件判断的另外一种形式，其功能与 If…Then…ElseIf 语句功能相同。在使用 If…Then…ElseIf 语句进行多条件判断时，可能会遇到需要将每个 ElseIf 块相同的表达式与不同的数值进行比较的情况，这样的结构在编写程序时就很枯燥，且程序不易阅读。

Select Case 语句的功能与 IF…Then…ElseIf 结构的功能类似，都是进行多条件的判断。但对于多重选择的情况，使用 Select Case 语句就能够大大地提高代码的编写效率，使代码的结构清晰且更易于阅读。Select Case 语句的语法格式如下：

```
Select Cass 测试表达式
    Case 值 1
        语句组 1
    Case  值 2
        语句组 2
    ...
    Case 值 n-1
        语句组(n-1)
    Case Else
        语句组 n
End Select
```

参数说明如下所示。

- 测试表达式：必需。用于条件判断。
- 值 n：条件入口参数。
- 语句组 n：可选，对应于值 n 所要执行的语句。

在 VBA 中，执行 Select Case 结构时，VBA 将计算测试表达式的值，然后将该值与每个 Case 的值进行比较。如果相等，就执行与 Case 相关联的语句组，执行完毕后转到 End Select 语句后，继续程序的执行。当有多个 Case 的值与测试表达式相匹配时，VBA 只执行第一个 Case 关联的语句组。如果没有任何一个 Case 的值与测试表达式相匹配，VBA 将执行 Case Else 的语句组。Select Case 结构的执行流程如图 6.13 所示。

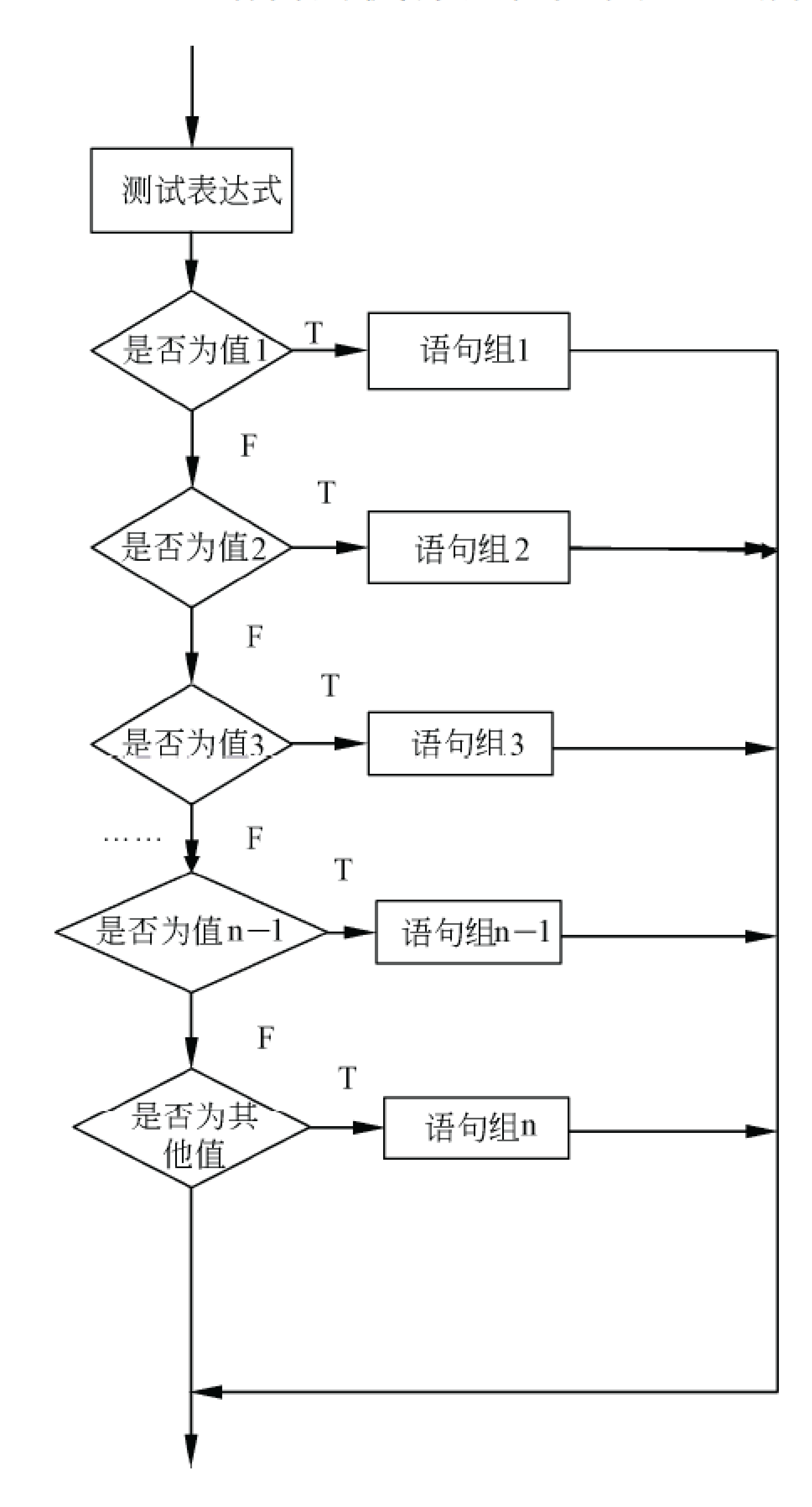


图 6.13　Select Case 语句流程

Select Case 结构中 Case 后面的值实际上是一个表达式列表，其可以是一个值也可以是几个值。如果是多个值，应该使用逗号（,）将其分隔开。Case 后面表达式的书写有下面几种情况。

- 使用表达式：这种方式用来表示一些具体的取值，如 Case 6,8,10。
- 表达式 1 To 表达式 2：这种方式用来表示一个数据取值范围，如 Case 5 To 8。
- 使用 Is 运算符：如 Case Is>100 表示所有大于 100 的数。
- 上述 3 种方式的混合：如 Case 0 To 10,20,Is>30，表示 0 至 10 间的数字、20 和所有大于 30 的数字。

注意：Select Case 中的测试表达式不能包含多个条件，这是与 If 语句不同的地方。Select Case 语句每次判断时都要使用开始处的表达式的值，而 If…Then…ElseIf 语句并不会每次都判断第一个 If 对应的表达式的值。因此，Select Case 语句只能理解为当 If 对应表达式和 ElseIf 表达式相同时的 If…Then…ElseIf 语句。

【范例 6-5】 使用 Select Case 语句将选择单元格中的数字转换为对应的月份，代码如下所示。

```
Sub 数字转换为月份()
    Dim a As Integer                              '声明变量
    a = Val(Selection.Value)                      '获得单元格数值
    Select Case a                                 '根据数值转换为月份
    Case 1                                        '数值为 1
        Selection.Value = "一月"                  '转换为“一月”
    Case 2                                        '数值为 2
        Selection.Value = "二月"                  '转化为“二月”
    Case 3                                        '数值为 3
        Selection.Value = "三月"                  '转换为“三月”
    Case 4                                        '数值为 4
        Selection.Value = "四月"                  '转换为“四月”
    Case 5                                        '数值为 5
        Selection.Value = "五月"                  '转换为“五月”
    Case 6                                        '数值为 6
        Selection.Value = "六月"                  '转换为“六月”
    Case 7                                        '数值为 7
        Selection.Value = "七月"                  '转换为“七月”
    Case 8                                        '数值为 8
        Selection.Value = "八月"                  '转换为“八月”
    Case 9                                        '数值为 9
        Selection.Value = "九月"                  '转换为“九月”
    Case 10                                       '数值为 10
        Selection.Value = "十月"                  '转换为“十月”
    Case 11                                       '数值为 11
        Selection.Value = "十一月"                '转换为“十一月”
    Case 12                                       '数值为 12
        Selection.Value = "十二月"                '转换为“十二月”
    Case Else                                     '数值非 1～12
        MsgBox "当前单元格中内容无法转换为月份"
        , vbOKOnly, "提示"                        '显示错误信息
    End Select
End Sub
```

【运行结果】创建一个工作表，如图 6.14 所示。切换到 Visual Basic 编辑器，在工程资源管理器中插入一个模块。在模块的“代码”窗口输入以上代码，按 F5 键运行程序，工作表中选择单元格的数字如果在 1～12 之间，则会被转换为月份，如图 6.15 所示。如果单元格中内容非 1～12 间的数字，程序给出提示，如图 6.16 所示。

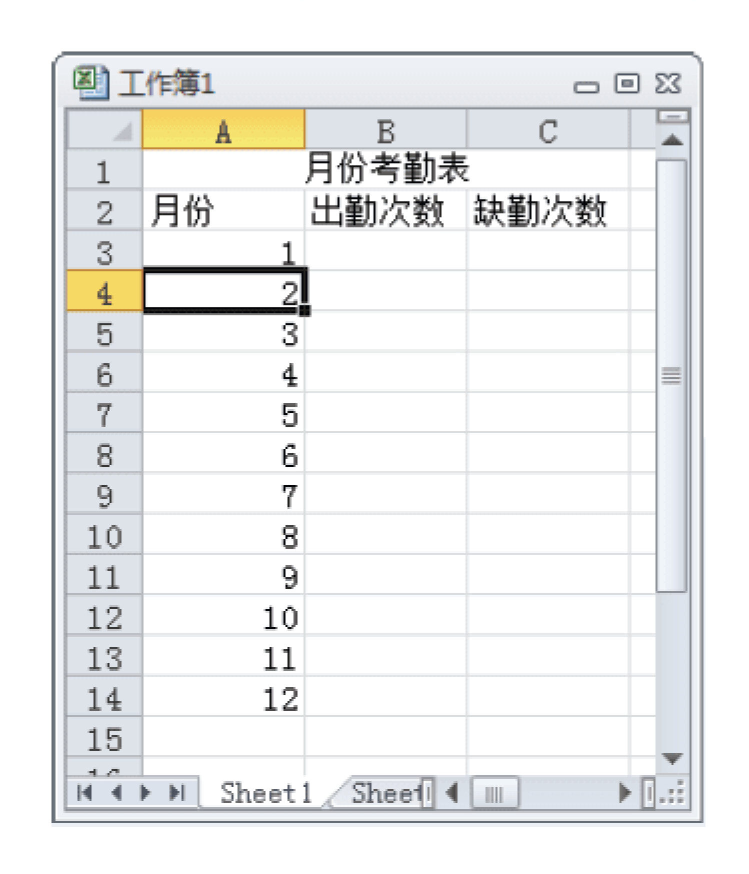


图 6.14　创建工作表

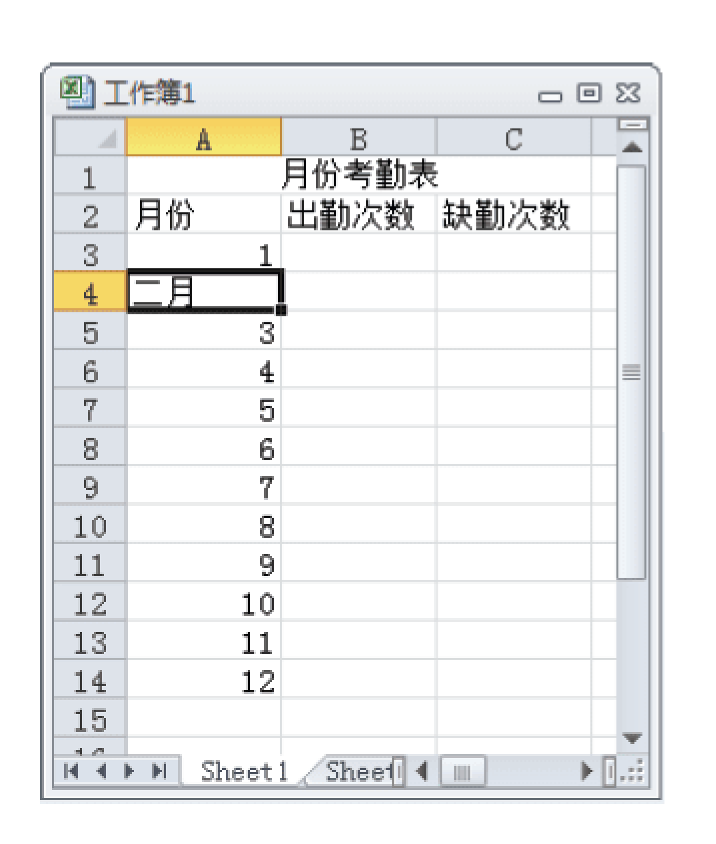


图 6.15　选择单元格中数字转换为月份

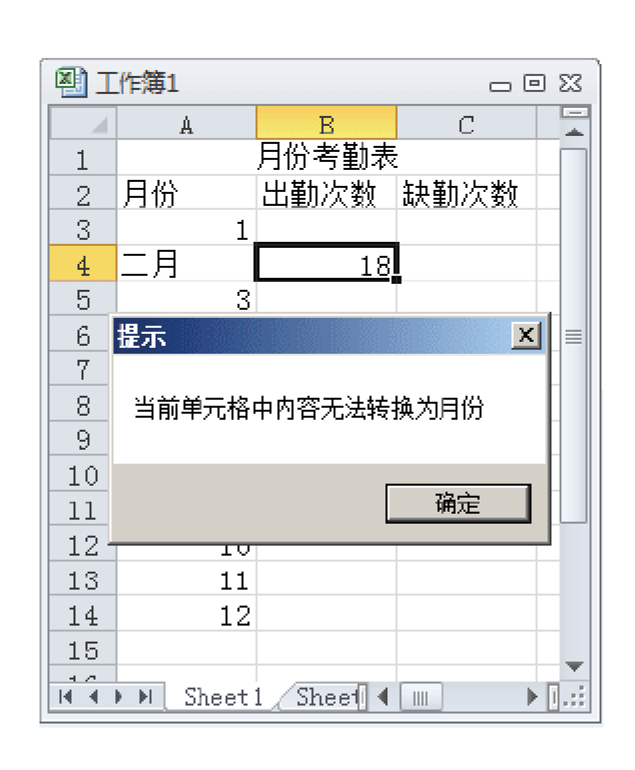


图 6.16　错误数据给出提示信息

【代码解析】本实例演示 Select Case 语句的使用方法。单元格中的数据是字符串型数据，因此在代码的第 03 行使用 Val 函数进行了数据的转换。程序以变量 a 的值作为条件，在代码的第 04～28 行，依次对数据的值进行判断，根据值的不同将数字转换为对应的月份。程序代码的第 29～31 行用于处理选定的单元格中的数字不是 1～12 间的数字时使用 MsgBox 函数给出提示信息。

6.1.8　被嵌套的选择结构

在一个选择结构中包含一个或多个选择结构语句，就构成了选择结构的嵌套。在程序中使用选择结构的嵌套可以实现对多个条件的选择，实现比单一选择结构更为复杂的逻辑选择。选择结构可以进行多层的嵌套，其语法结构如下所示。

```
If<表达式 1>Then
    If<表达式 2>Then
        [语句组 1]
Else
        [语句组 2]
End If
    Else
    If<表达式 3>Then
        [语句组 3]
Else
        [语句组 4]
    End If
End If
```

【范例 6-6】 使用输入框向工作表添加日期，输入时对输入框中的输入进行判断，代码如下所示。

```
Sub 选择结构的嵌套()
    Dim d As Variant                                    '声明变量
    d = InputBox("请输入日期: ", "输入日期")              '输入框输入日期
    If d <> "" Then                                     '判断是否输入
        If IsDate(d) Then                               '判断输入是否为日期
            Selection.Value = d                         '日期被写入选择单元格
        Else
            MsgBox "输入格式不对，非日期！"                '提示输入非日期
        End If
    Else
        MsgBox "输入为空！"                               '提示输入为空
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，在输入对话框中输入日期，单击“确定”按钮关闭对话框后，日期被写入选择单元格中，如图 6.17 所示。如果未输入日期即关闭对话框，程序给出提示，如图 6.18 所示。

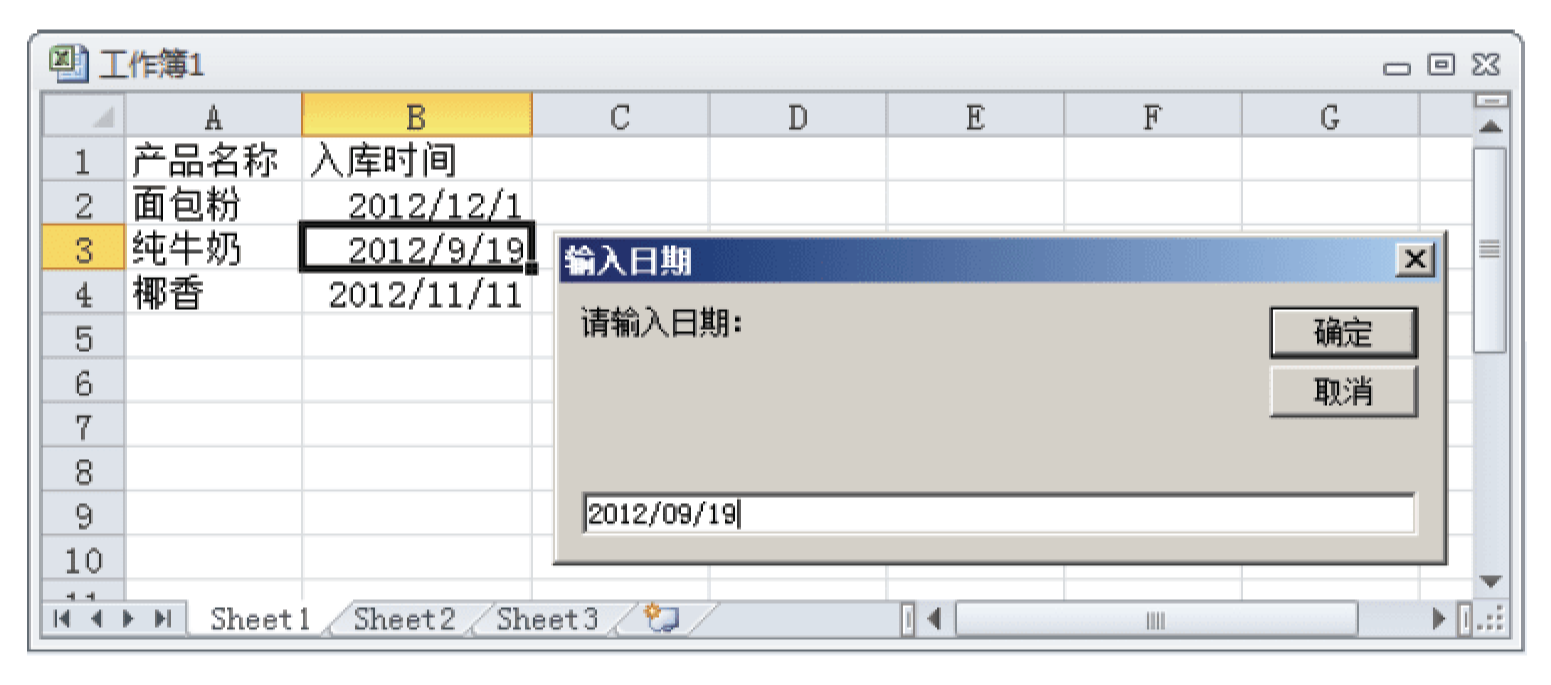


图 6.17　输入对话框中输入日期后关闭对话框日期并写入选择单元格



图 6.18　输入框中未字符时的提示

如果输入非日期，程序给出提示，如图 6.19 所示。



图 6.19　输入给日期时给出的提示

【代码解析】这段代码使用了 If 语句的嵌套来实现对多重条件的判断。代码比较简单，首先检验是否输入，即判断 d 的值是否为空。如果有输入，则使用 IsDate 函数检验输入值是否是日期，如果是有效日期，则将日期写入工作表中选择的单元格中。否则给出提示，本示例的流程如图 6.20 所示。

🔔注意：在使用 If 语句时嵌套时，每一个 If 语句必须对应有一个 End If 语句来与之对应，否则程序运行时就会提示错误。在进行多层嵌套时，使用缩进将有利于理解嵌套的层次关系。

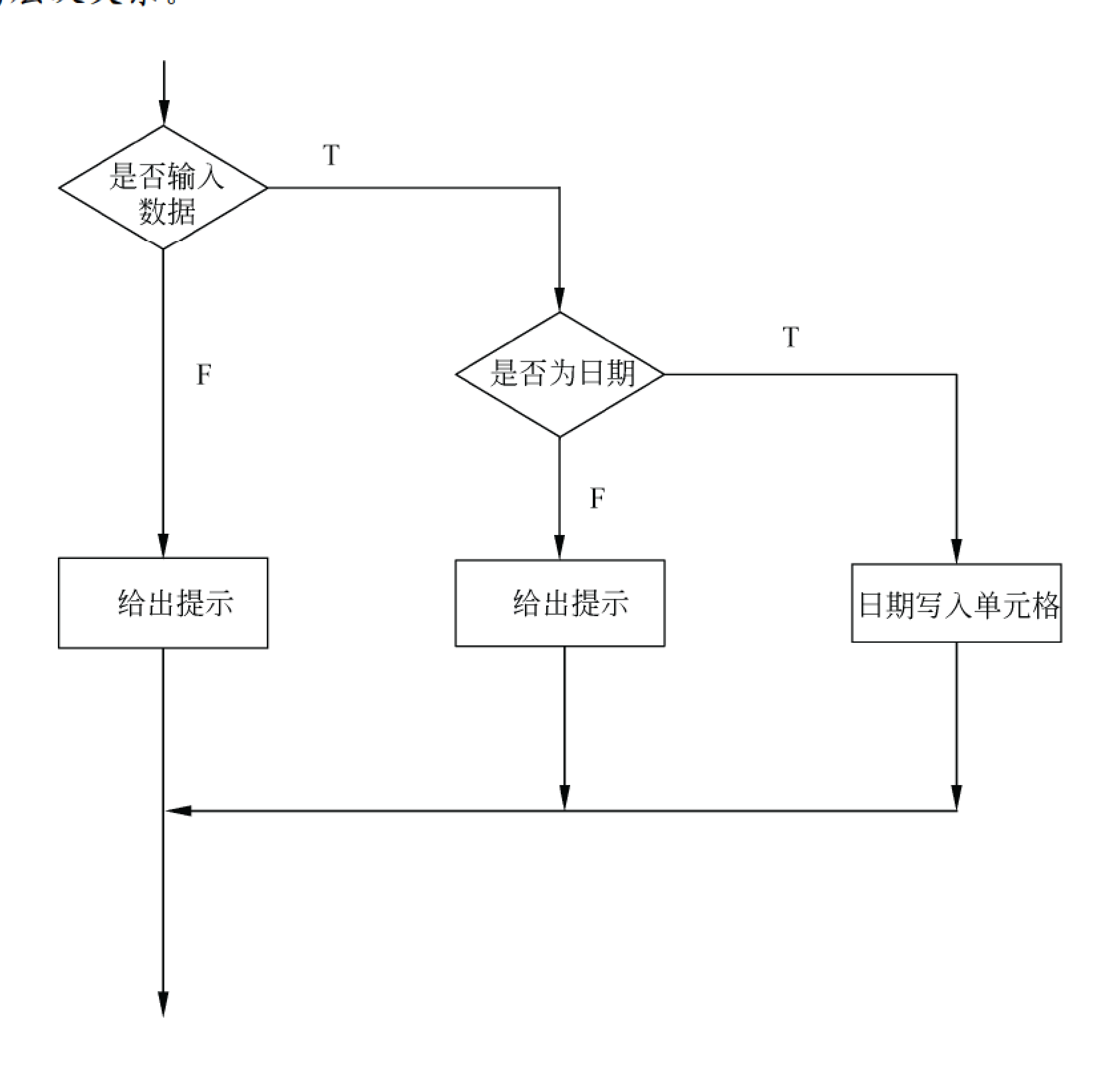


图 6.20　示例流程图

6.2　使用 VBA 循环结构

在程序设计过程中，经常会碰到需要按照一定的规律重复执行的运算或操作。VBA 的循环结构就是用来实现某段程序的反复执行的，这种用来实现程序循环执行的语句称为循环语句，其中被重复执行的语句称为循环体。本章将对 VBA 的循环语句进行介绍。

6.2.1　For…Next 语句

For…Next 语句用于执行给定次数的循环，即在给定次数内，循环体内的程序将会被反复执行，直到达到设定的计数次数，循环体被执行的循环次数是由称为循环计数器的变量决定。当循环每执行一次时，循环计数器变量的值自动增加或减少。当达到规定的值时，循环停止。For…Next 语句的语法格式如下：

```
For 循环变量=初始值 To 终值[Step<步长>]
    语句组 1
    [Exit For]
```

```
    [语句组 2]
Next [循环变量]
```

参数说明如下所示。

- 循环变量：必需。设定循环次数的计数变量名。
- 初始值：必需。循环变量的初始值。
- 终值：必需。循环变量的结束值。
- 步长：可选，变量每次循环后变化的数值。默认值为 1。
- 语句组 1：即循环结构的循环体。由一条或多条语句组成，完成所要实现的功能。
- 语句组 2：可选，功能与语句组 1 相同。
- Exit For：可选，用于直接退出循环体，执行 Next 后的语句。

For…Next 语句在执行时，当步长值为正数时，如果初始值小于或等于终止值时，循环体将执行，否则将退出循环。如果步长为负数，则初始值必须大于或等于终止值循环体才能执行。在没有设置步长时，步长的默认值为 1。For…Next 语句的流程如图 6.21 所示。

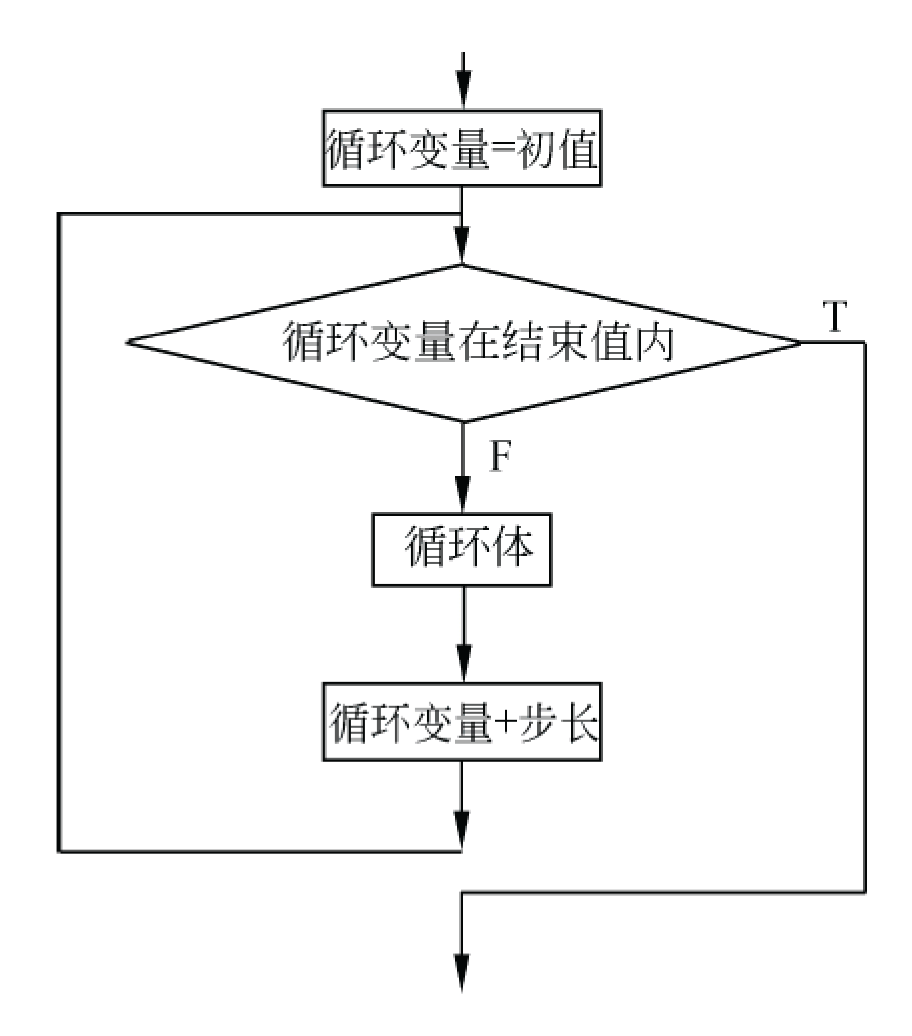


图 6.21　For…Next 语句的流程图

【范例 6-7】 使用 For…Next 循环，删除工作表前 10 行中的没有记录信息的行，代码如下所示。

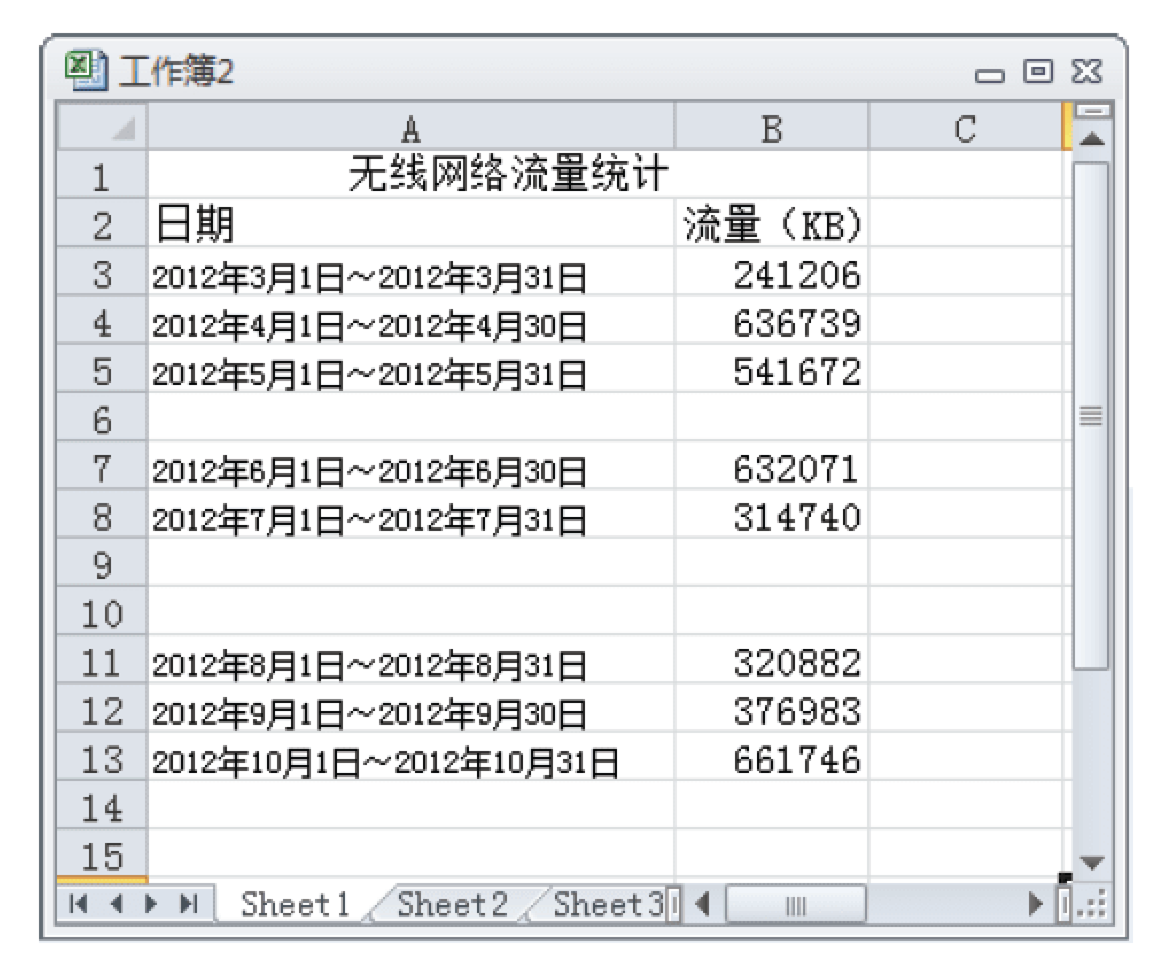
```
Sub 循环语句示例()
    Dim n As Integer                          '声明计数变量
    For n = 1 To 10                           '从 1 到 10 循环
        If Cells(n, 1) = "" Then              '如果单元格为空
            Cells(n, 1).EntireRow.Delete      '删除单元格所在行
        End If
    Next
End Sub
```

【运行结果】启动 Excel 2010，创建工作表，工作表中存在着输入不全的行，例如记录信息没有输入，如图 6.22 所示。切换到 Visual Basic 编辑器，插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序运行后将把工作表第 1~10 行中没有记录信息的行删除。程序运行后的效果，如图 6.23 所示。

【代码解析】这段代码使用 For…Next 循环来对指定单元格进行遍历。计数变量从 1 到

10，循环体中的 Cells（n，1）指定单元格为第 1 列的第 1~10 个单元格。程序中使用 If 语句判断单元格是否为空，如果为空，Cells(n,1).EntireRow 指定的行使用 Delete 方法删除。

提示： For…Next 循环中循环体被执行的次数可以使用下面这个公式计算出来：循环次数 = [（终止值 – 初始值）/步长值]+1，这里计算的结果应该取正数。另外要注意，循环变量在循环中的作用除了控制循环次数外，还可以为循环中的变量或属性赋值。



	A	B	C
1	无线网络流量统计		
2	日期	流量（KB）	
3	2012年3月1日~2012年3月31日	241206	
4	2012年4月1日~2012年4月30日	636739	
5	2012年5月1日~2012年5月31日	541672	
6			
7	2012年6月1日~2012年6月30日	632071	
8	2012年7月1日~2012年7月31日	314740	
9			
10			
11	2012年8月1日~2012年8月31日	320882	
12	2012年9月1日~2012年9月30日	376983	
13	2012年10月1日~2012年10月31日	661746	
14			
15			

图 6.22　存在没有输入记录信息的行

	A	B	C
1	无线网络流量统计		
2	日期	流量（KB）	
3	2012年3月1日~2012年3月31日	241206	
4	2012年4月1日~2012年4月30日	636739	
5	2012年5月1日~2012年5月31日	541672	
6	2012年6月1日~2012年6月30日	632071	
7	2012年7月1日~2012年7月31日	314740	
8	2012年8月1日~2012年8月31日	320882	
9	2012年9月1日~2012年9月30日	376983	
10	2012年10月1日~2012年10月31日	661746	
11			
12			

图 6.23　删除没有记录信息的行

6.2.2　For Each…In Next 语句

For Each…In Next 语句用于对数组或者集合中每个元素进行循环操作，使用该语句能够遍历一个确定的集合中的每个对象。在编写 VBA 程序时，往往会遇到无法确定循环次数的情况，此时无法使用 For…Next 语句进行计数循环，只能使用 For Each…In Next 语句来实现循环操作。For Each…In Next 实现语句的语法结构如下：

```
For Each 变量 In 数组或集合
    语句组
Next 变量
```

参数说明如下所示。

- ❑ 变量：必要。用来遍历数组或集合中的所有元素。对于数组，这个变量只能是一个 Variant 类型变量。对于集合，可能是一个 Variant 类型变量、一个通用对象变量或者是任何对象变量。
- ❑ 数组或集合：必需。数组或集合的名称。
- ❑ 语句组：可选，由一条或多条语句组成，完成所要实现的功能。

【范例 6-8】 创建一个程序统计成绩表中大于 80 分的分数的个数，同时将大于 80 的分数用颜色标注出来，代码如下所示。

```
Sub 统计成绩高于 80 的人数()
    Dim rng As Range, n As Integer                    '声明单元格集合和变量
```

```
03          For Each rng In Range("b3:c18")          '遍历单元格
04              If rng >= 80 Then                     '判断分数是否大于等于 80
05                  n = n + 1                         '计数增加 1
06                  rng.Interior.ColorIndex = 40      '设置单元格颜色
07              End If
08          Next rng
09      MsgBox "分数超过 80 分的一共有" & n & "人次。", _
10       vbOKOnly, "统计结果"                          '显示总数
11  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序会为大于 80 分的单元格填充颜色，同时显示统计结果，如图 6.24 所示。

【代码解析】本段代码使用 For Each…In Next 语句循环结构遍历工作表中指定的单元格。在代码的第 01 行声明了一个单元格对象变量，声明该对象后就能像 Excel 单元格对象那样使用它了。代码中使用 For Each…In Next 循环遍历 B3 至 C18 区域中的每个单元格，判断区域中每个单元格的值是否大于等于 80，对其中数值大于等于 80 的单元格计数并填充颜色。在代码中使用 Interior 对象的 ColorIndex 属性来设置符合条件单元格的颜色。

注意：在 For Each…In Next 循环中，作为变量的元素所声明的变量类型一定要属于集合的元素类型，如示例中集合是 Range，则变量必须声明为 Range 类型。如果事先无法知道集合类型，可将其声明为变体型变量。

图 6.24　程序运行的结果

6.2.3　Do…Loop 语句

使用 For…Next 来实现循环，循环的次数必须能够确定。但在程序设计中经常会遇到循环次数无法确定的情况，如，在对学生成绩进行统计时，需要逐个查询学生的成绩，将及格的学生挑选出来。此时，很难确定及格学生的具体人数，显然无法使用 For…Next 循环来实现操作。

Do…Loop 循环是一种根据条件的真假来决定循环是否继续的循环结构，其特别适合上面介绍的那种不能使用计数器来控制循环次数的场合。Do…Loop 型循环结构有两种情况，它们是 While 型的 Do…Loop 循环和 Until 型的 Do…Loop 循环。本节将首先介绍 While 型 Do…Loop 循环。

While 型 Do…Loop 语句用于重复执行一组语句。当 While 后的表达式结果为 True 时，执行循环体，直到表达式值为 False 时退出循环。根据 While 语句位置的不同 While 型 Do…Loop 语句分为 Do While...Loop 结构和 Do...Loop While 结构这两种情况，其结构如图 6.25 和图 6.26 所示。

Do While...Loop 结构和 Do...Loop While 结构的语法格式分别如下所示。

```
Do While 表达式
语句组
[Exit Do]
[语句组]
Loop
```

或

```
Do
语句组
[Exit Do]
[语句组]
Loop While 表达式
```

参数说明如下所示。

- ❑ 表达式：可选，与 While 关键字配合判断循环条件。
- ❑ 语句组：可选，一条或多条语句，执行所要完成的功能。
- ❑ Exit Do：可选，退出循环。

以上两种语法格式的区别在于，Do While...Loop 语句在进入循环体时，先判断表达式结果。如果表达式结果为 True 则执行循环体，如果表达式结果为 False 则跳出循环。Do...Loop While 语句首先执行一次循环体，然后再判断表达式结果。也就是说，如果完成相同的任务，表达式结果为 False 时，Do While...Loop 语句不执行循环体退出，而 Do...Loop While 语句执行一次循环体后退出。

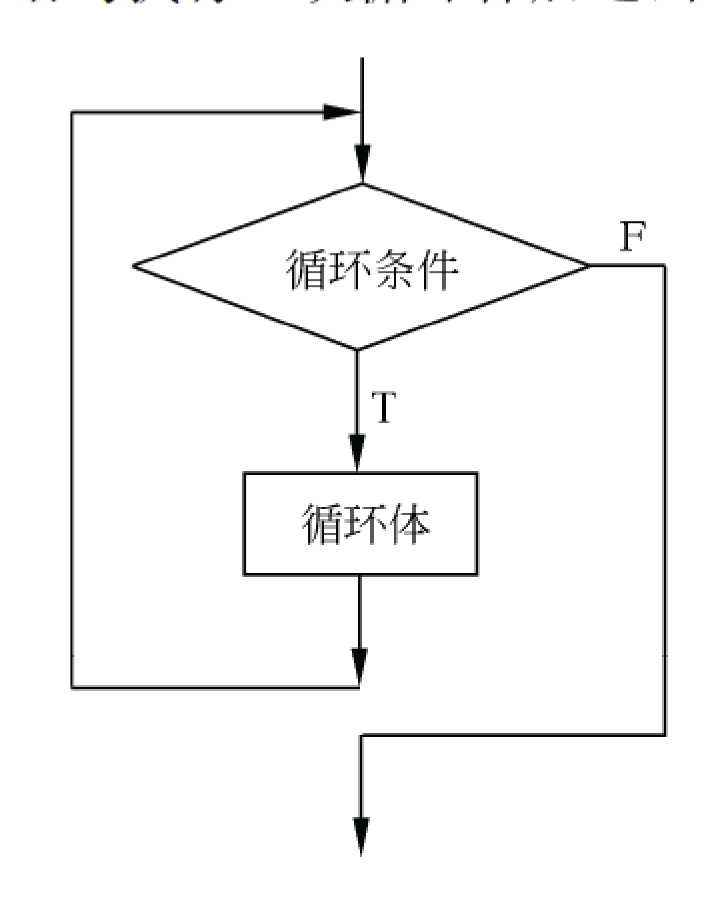


图 6.25　Do While...Loop 语句流程图

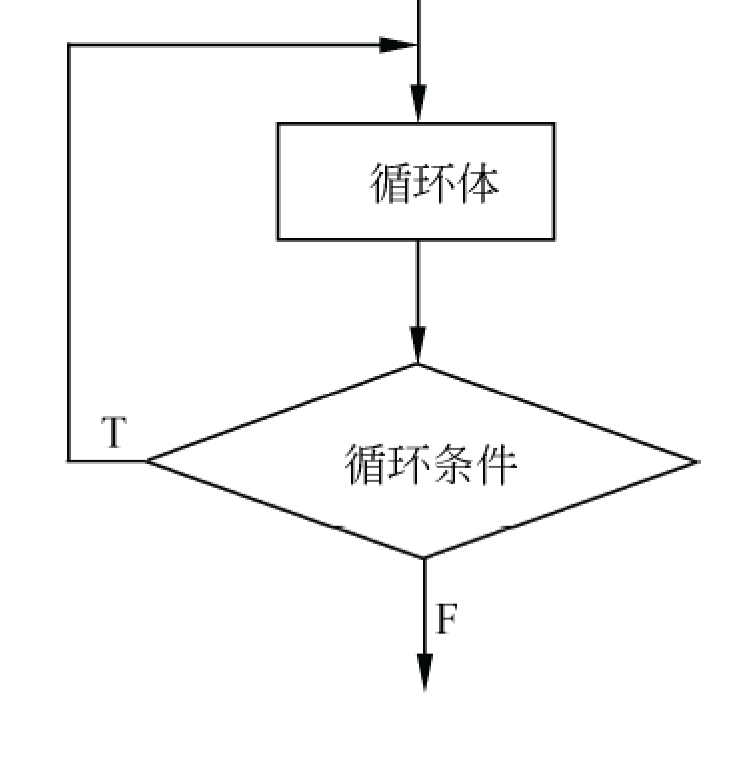


图 6.26　Do...Loop While 语句流程图

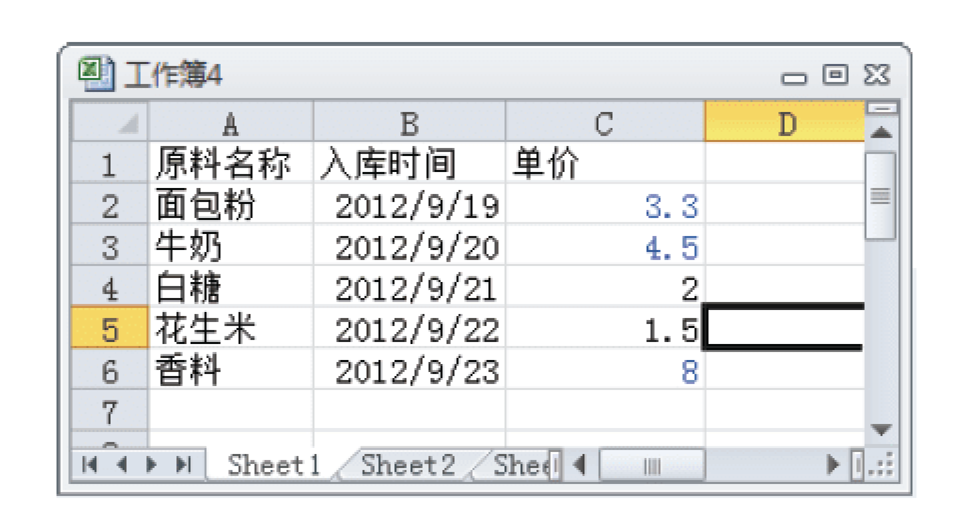
【范例 6-9】 使用 Do While...Loop 循环查阅工作表中产品价格，使价格高于 3 元的数据显示为蓝色，代码如下所示。

```
Sub 向单元格中输入数字()
    Dim n As Integer                            '声明计数变量
    n = 1                                       '变量初始化
    Do While n < 7                              '开始循环
        n = n + 1                               '计数变量加 1
        If Cells(n, 3) > 3 Then                 '判断值是否大于 3
            Cells(n, 3).Font.ColorIndex = 5     '大于 3 的值改变颜色
        End If
    Loop
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，工作表中单价高于 3 元的价格数据被显示为蓝色，如图 6.27 所示。

【代码解析】本段代码可以演示使用 Do While...Loop 语句遍历指定单元格的方法。在

程序中，首先初始化计数变量 n，循环的条件是 n<1。在循环体中，必须使用赋值语句将循环变量加 1，这是与 For..Next 循环不同的地方。在循环体中使用 If 语句判断单元格中数据与 3 的大小关系，如果大于 3，使用 Cells(n, 3).Font.ColorIndex = 5 语句将文字的颜色设置为蓝色。



	A	B	C	D
1	原料名称	入库时间	单价	
2	面包粉	2012/9/19	3.3	
3	牛奶	2012/9/20	4.5	
4	白糖	2012/9/21	2	
5	花生米	2012/9/22	1.5	
6	香料	2012/9/23	8	
7				

图 6.27　程序运行的效果

注意：在使用 Do ...Loop 语句创建循环结构时，提倡使用 Do While...Loop 语句，这样可以避免代码中出现不必要的逻辑错误。

6.2.4　Until 型 Do…Loop 语句

与 While 型 Do…Loop 语句相反，Until 型 Do…Loop 语句执行循环体，直到 Until 后的表达式结果为 True 时退出循环，而当 Until 后的表达式结果为 False 时执行循环体。根据 Until 语句位置的不同，Until 型 Do…Loop 语句包含有两种类型，它们是 Do Until…Loop 语句和 Do…Loop Until 语句。这两种循环结构的流程图如图 6.28 和图 6.29 所示。

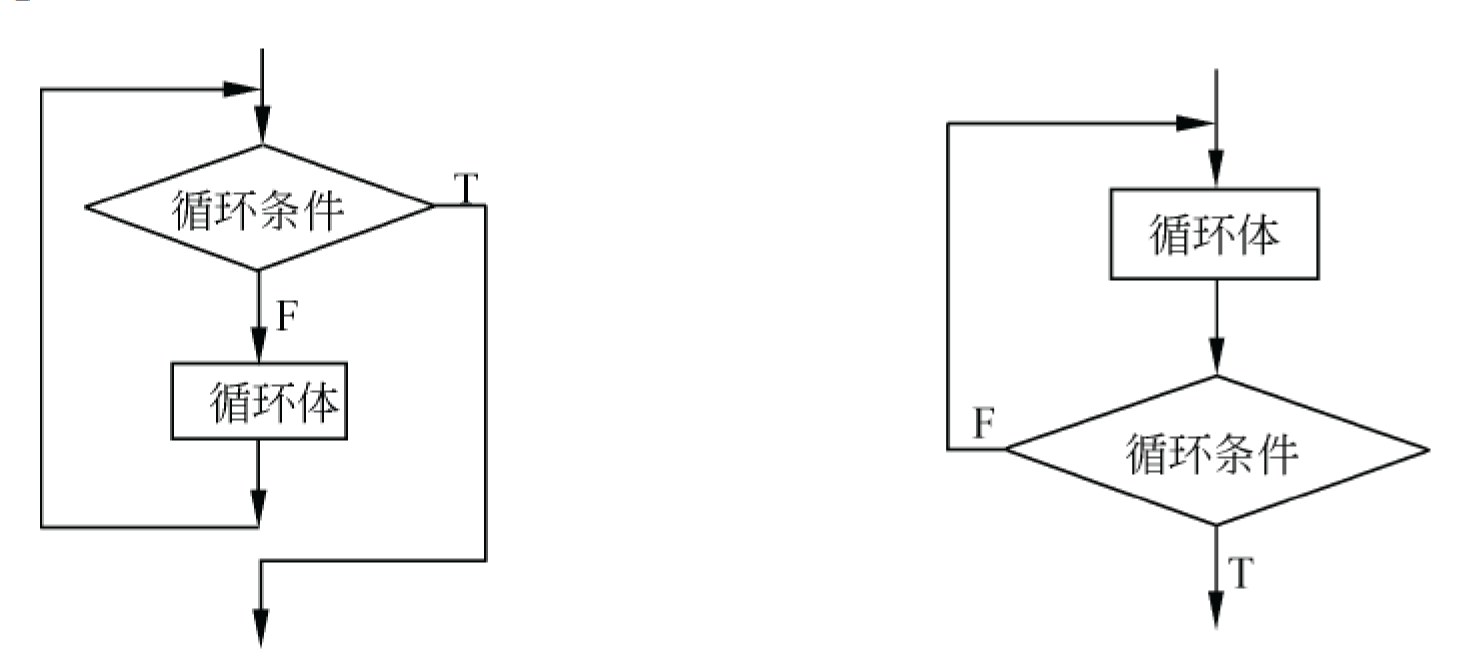


图 6.28　Do Until…Loop 语句流程图　　图 6.29　Do…Loop Until 语句流程图

上面两种 Until 型 Do…Loop 语句的语法结构如下所示。

```
Do Until 表达式
语句组
[Exit Do]
Loop
```

或

```
Do
语句组
[Exit Do]
```

```
Loop Until 表达式
```

参数说明如下所示。

- 表达式：可选，与 Until 关键字配合判断循环条件。
- 语句组：可选，一条或多条语句，执行所要完成的功能。
- Exit Do：可选，退出循环。

【范例 6-10】 使用 Until 型 Do…Loop 语句解决预算问题。某工厂现在的年产值为 100 万元，按年增长率 8%计算，多少年后其年产值超过 140 万元。解决此问题的代码如下所示。

```
Sub 增长率的问题()
    Dim a As Integer, x As Single    '声明变量
    n = 0                            变量初始化
    x = 100
    Do                               '使用 Until 型 Do…Loop 语句循环计算
        x = x * 1.08
        n = n + 1                    '循环计数
    Loop Until x > 140
    Debug.Print n                    '显示满足条件的 n 的值
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，在“立即窗口”中显示此题的解，如图 6.30 所示。



图 6.30　“立即窗口”显示此题的解

【代码解析】这段代码用于演示 Do ...Loop Until 循环语句的使用方法。在代码中，根据题目给出的条件，计算每一年增长 8%后的年产值，使用 Do…Loop Until 来实现循环的控制，即当 x 值大于了 140，循环停止。在循环体中，使用赋值语句 n=n+1 来计算循环次数，这个次数就是年度值。

提示： Do ...Loop Until 循环语句，其中的循环体至少可以执行一次。而 Do Until…Loop 语句中的循环体可能执行多次，也有可能一次都不会执行，这是由变量值所决定的。

6.2.5　While…Wend 语句

While…Wend 语句用于对条件进行判断，在程序运行时，如果条件结果为 True，就会重复执行循环体内的语句组。否则，将跳出循环体转入执行 Wend 后的语句。While…Wend 语句流程如图 6.31 所示。

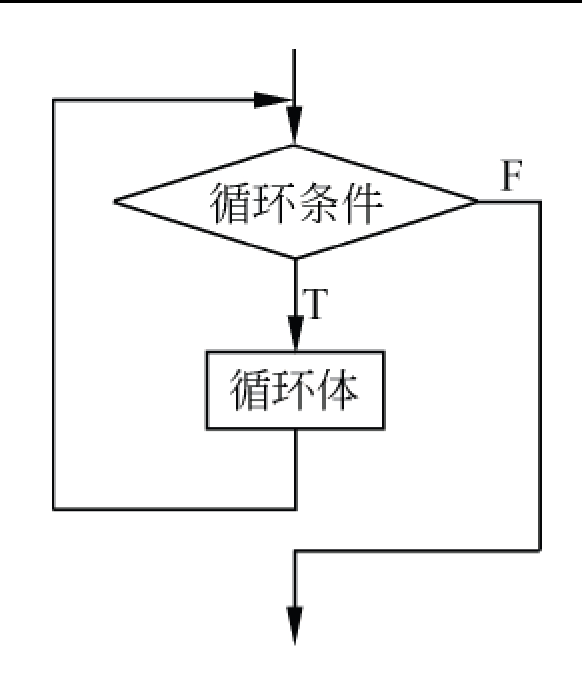


图 6.31　While…Wend 语句流程图

While…Wend 语句与 Do…While 语句功能相同，其语法结构如下：

```
While 表达式
语句组
Wend
```

参数说明如下所示。

- ❑ 表达式：必需。表达式用于给出条件结果。
- ❑ 语句组：可选，由一条或多条语句组成，完成所要实现的功能。

【范例 6-11】 使用 While…Wend 循环来制作一个限制输入次数的密码输入框。程序运行时要求在输入框中输入密码，密码输入次数限制为 3 次，程序代码如代码示例 6-11 所示。

```
Sub 应用 While…Wend 语句()
    Dim p As Variant                                    '声明变量
    Dim i As Integer
    i = 0                                               '计数变量初始化
    While i < 3 '小于 3 时循环
        i = i + 1 '计数变量加 1
        p = InputBox("请输入密码：")                    '输入对话框中输入密码
        If p = "1234" Then '验证密码
            MsgBox "密码输入正确！"                     '提示密码输入正确
            Exit Sub                                    '退出过程
        Else
            MsgBox "密码输入错误，请重新输入！"         '提示密码错误
        End If
    Wend
    MsgBox "密码输入超过 3 次！"                        '提示密码输入超过 3 次
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，在输入对话框中输入密码，如图 6.32 所示。如果密码正确，程序给出提示后退出过程，如图 6.33 所示。



图 6.32　输入密码



图 6.33　密码正确提示

如果输入密码错误，程序给出提示，如图 6.34 所示，当输入密码错误超过 3 次时，程序将给出提示，同时退出过程，如图 6.35 所示。



图 6.34　密码输入错误提示



图 6.35　密码错误超过 3 次时的提示

【代码解析】本示例演示使用 While…Wend 循环来实现输入对话框的输入限制。程序中计数变量的初始值为 0，循环条件是 i<3。每次循环都使用 InputBox 函数来获取用户输入密码，将输入值与设定的密码比较，如果相符，则给出提示并退出过程。如果不相符则给出提示，循环继续进行。当循环次数超过 3 次时，退出循环并给出超过输入次数的提示。

警告：While…Wend 语句的关键在于判断条件，读者在使用 While…Wend 语句编写循环结构时一定要注意判断条件是否正确或符合逻辑。

6.2.6　被嵌套的循环结构

在编写 VBA 应用程序时，很多问题用单循环也不能有效地解决，此时可以通过循环语句的嵌套来解决。所谓循环的嵌套是指在程序结构将一个或多个循环结构放置在另一个循环结构中。这种嵌套结构在执行时，外层的循环（即外循环）每执行一次，嵌套在其中的内层循环结构（即内循环）就会从头开始执行一轮。

在使用循环嵌套时，内循环结构必须完整地被外循环结构包含。同时，内循环结构不能和外循环结构相互交叉。内循环结构和外循环结构的变量名必须是唯一的（不能重复）。

【范例 6-12】 使用 For...Next 构成两层循环嵌套，统计工作表已检测的产品的个数，程序代码如下所示。

```
Sub 循环嵌套的应用()
    Dim n As Integer, i As Integer, c As Integer   '声明变量
    For n = 1 To Range("a1048576").End(xlUp).Row   '遍历每一行
        For i = 1 To 3                             '遍历每一列
            If Cells(n, i) = "是" Then c = c + 1
                                       '单元格内容为“是”，变量加 1
        Next
    Next
    MsgBox "共有" & c & "件产品已被检验。"        '显示获得的单元格个数
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序提示工作表中标记为检验产品个数，如图 6.36 所示。

【代码解析】本示例代码使用了两层循环的嵌套来遍历工作表中单元格，查找其中标记为“是”的单元格并统计个数。在程序中，Range("a1048576").End(xlUp).Row 获得 A 列最后一个非空单元格的行号。以其作为上限，使用 For...Next 循环遍历所有 A 列非空的行。

第 04 行代码至第 06 行循环嵌套在第一层循环中，遍历每一行的前 3 列的每一个单元格，判断其内容是否为“是”，如果是则计数。完成循环后，在提示对话框中显示统计结果。

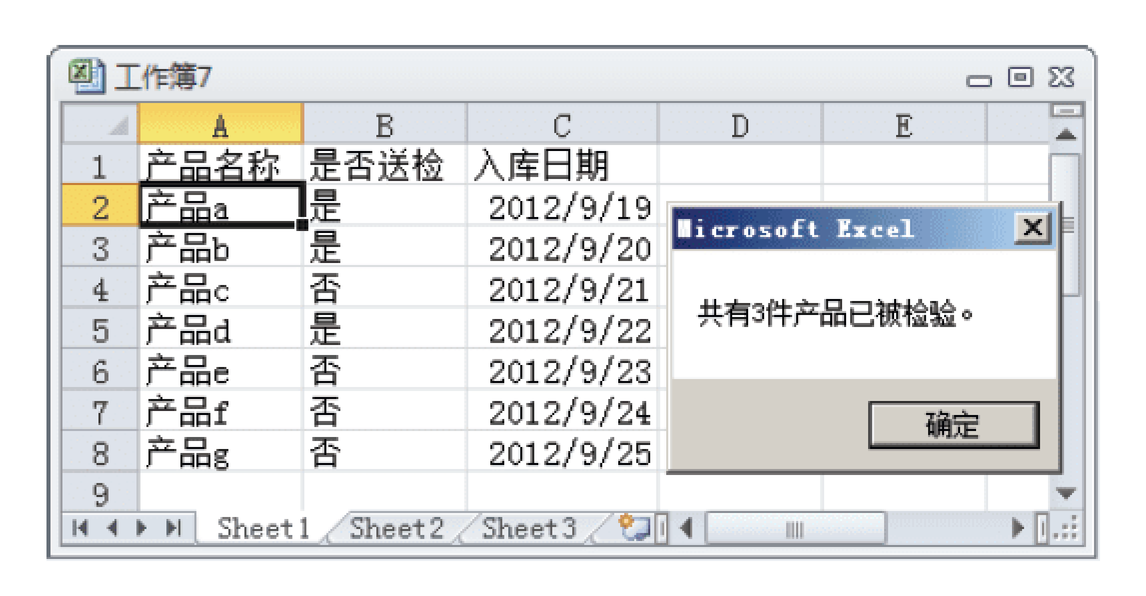


图 6.36　程序运行效果

6.3　使用其他控制语句

除了以上介绍的结构控制语句，VBA 还提供了 GoTo 语句、Exit 语句、End 语句、With 语句以及 DoEvents 语句来完成程序流向的控制。本节将主要对 GoTo 语句、With 语句和 Exit 语句进行介绍。

6.3.1　With 语句

With 语句用于定制一个对象或读者自定义类型。在编写程序时，如果需要设置一个对象的多个属性就可以使用 With 语句。在程序中使用 With 语句能够提高程序运行效率，方便读者定制对象，增强代码的可读性，使得程序结构更加清晰。With 语句的语法如下：

```
With 对象
    .属性 1=属性值
    .属性 2=属性值
End With
```

参数说明如下所示。

- ❑ 对象：必需。一个对象或读者自定义类型。
- ❑ 属性 1 和属性 2：同一个对象的不同的属性。
- ❑ 属性值：需要分别赋予对象属性 1 和对象属性 2 的值。

【范例 6-13】 使用 With 语句更改选择单元格填充颜色和文字样式，以对单元格数据进行强调，代码如下所示。

```
Sub 设置对象属性()
    With Selection
        .Font.Bold = True                        '设置文字为加粗样式
        .Font.Size = 15                          '设置文字大小
        .Font.Name = "隶书"                      '设置文字字体
        .Font.Italic = True                      '设置文字为倾斜样式
        .Interior.Color = RGB(255, 255, 0)       '设置单元格填充颜色为黄色
    End With
```

```
09  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。在工作表中选择单元格后运行程序，单元格中填入指定样式的文字，单元格被填充颜色，如图 6.37 所示。

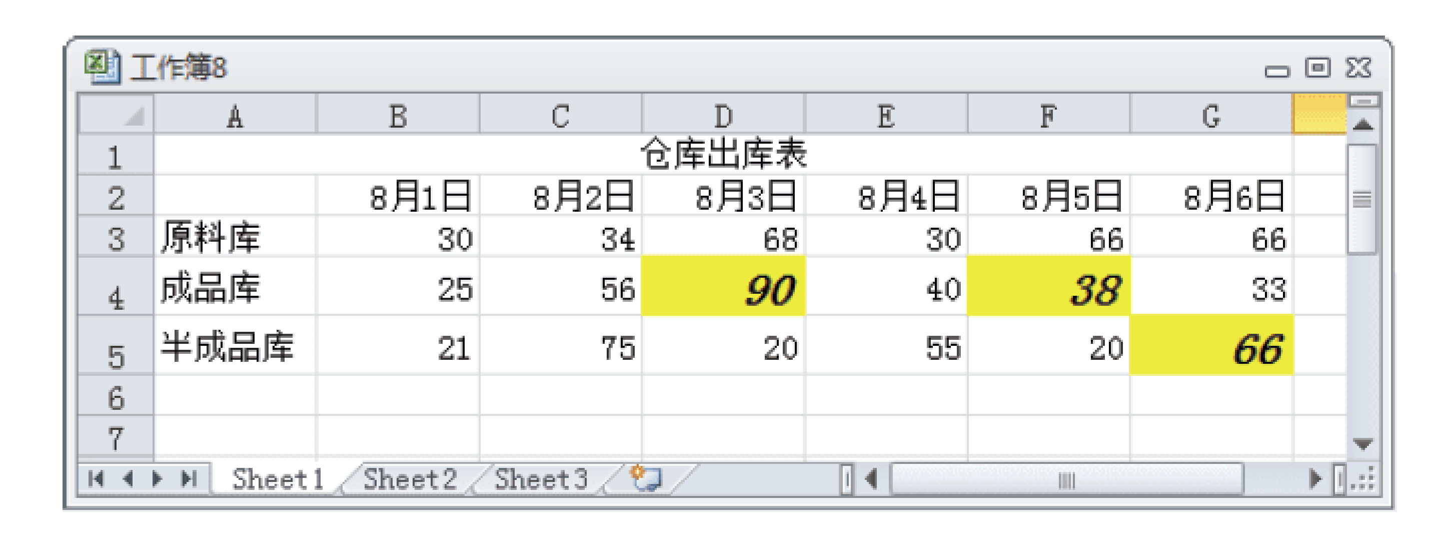


图 6.37　程序运行效果

【代码解析】在这段代码中，使用 With 结构同时设置 Selection 对象的多个属性，这些属性包括对象填充内容和对象填充的颜色，以及对象中文字的字体、字号以及加粗等属性。在代码中使用 RGB 函数来生成颜色值，RGB 函数的语法结构为 RGB（r，g，b），其中的参数 r、g 和 b 分别表示颜色中红色、绿色和蓝色颜色成分，其取值均为 0～255。

注意：在 With 结构中，结束必须要有 End With 语句，否则程序将会出错。另外，With 结构可以像循环结构那样进行嵌套以实现对多个对象的操作。

6.3.2　Exit 语句

在程序中，有时需要直接退出当前正在执行的循环或模块，此时可以使用 Exit 语句来实现。Exit 语句用于退出 For...Next、Do...Loop、Sub、Function 以及 Property 代码块。当需要退出 For…Next 和 For Each…In Next 循环时，可以在循环体中添加如下语句。

```
Exit For
```

如果需要退出 Do…Loop 循环并执行 Loop 语句之后的程序，可以在循环体中添加下面语句。

```
Exit Do
```

当需要退出 Sub 过程，并执行 Sub 过程后的语句，可以在 Sub 过程中添加下面的语句。

```
Exit Sub
```

当需要退出 Function 过程，同时程序将执行 Function 后的语句，可以在 Function 过程中添加下面语句。

```
Exit Function
```

如果需要退出 Property 代码块，同时程序执行 Property 后的语句，可以在 Property 代码块中添加如下语句。

```
Exit Property
```

【范例 6-14】 设计一个成绩录入系统，要求使用输入对话框输入成绩，输入成绩填入

工作表的单元格中。单击输入对话框的“取消”按钮退出输入，并给出退出提示，代码如下所示。

```
01  Sub 输入分数()
02      Dim a As Integer, b As Integer                     '声明变量
03      b = 2                                              '变量初始化
04      Do While True                                      '进入循环
05          a = Val(InputBox("请录入成绩", "成绩录入系统"))'录入分数
06          If a = False Then Exit Do                      '按取消键则退出录入
07          Cells(b, 2) = a                                '录入数字写入单元格
08          b = b + 1                                      '变量加 1 指向下一行
09      Loop
10      MsgBox "录入结束，回到工作表查看录入的成绩", _
11       vbOKOnly, "提示"                                  '显示退出提示信息
12  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，屏幕上显示“成绩录入系统”对话框。输入成绩后单击“确定”按钮，成绩填入单元格中，同时再次出现相同的对话框要求输入下一个成绩，如图 6.38 所示。单击“成绩录入系统”对话框中的“取消”按钮将退出成绩的录入并给出提示，如图 6.39 所示。

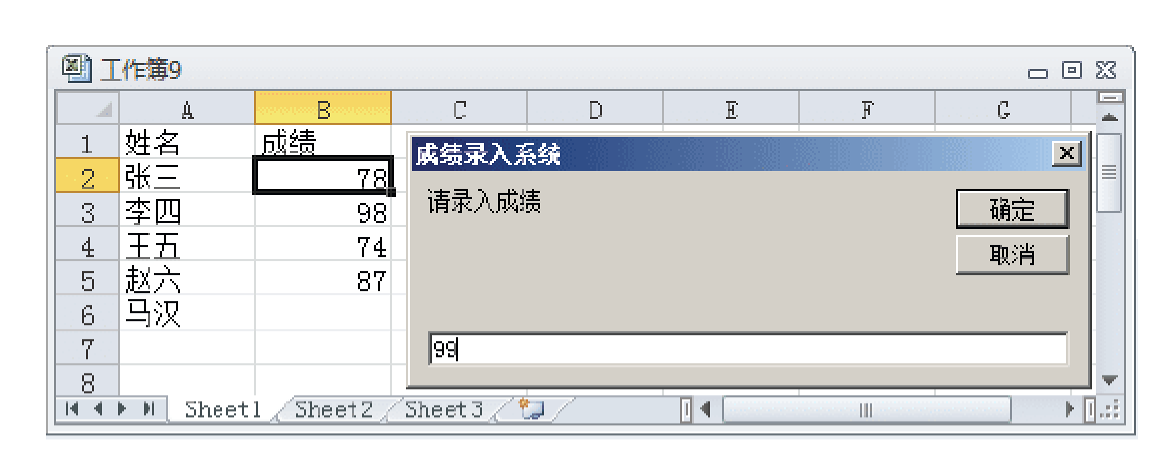


图 6.38　向工作表中录入成绩



图 6.39　退出提示

【代码解析】本实例成绩的录入需要反复进行，使用 Do While…Loop 循环来实现输入对话框的反复出现。这里，循环条件设置为 True，以此作为条件的循环将会无限制地进行下去，直到执行 Exit 语句退出循环。第 06 行代码对变量 a 的值进行判断，如果为 False，使用 Exit Do 语句结束当前循环。退出循环后，程序将执行循环结构后面的语句，即给出提示对话框。

注意：InputBox 函数在单击“确定”按钮时将返回文本框中的输入值，当单击“取消”按钮时，返回值为 False。

6.3.3　GoTo 语句

GoTo 语句用于程序无条件地跳转到指定的语句并执行该语句。GoTo 语句语法如下所示。

```
GoTo 行号|行标签
```

参数：为 GoTo 语句跳转提供目的行标识。行标签以“：”结尾。

注意：随着结构化程序设计方法的广泛使用，程序中已经很少为每行代码添加行号了。为了使用 GoTo 语句跳转到程序中需要的位置，常常采用在程序中添加行标签的方式。

【范例 6-15】 计算学生成绩表中选择单元格所在行的测试成绩平均分，如果成绩空白，给出提示，代码如下所示。

```
01  Sub 程序跳转()
02      Dim c As Integer, q As Integer, p As Integer   '声明变量
03      q = Selection.Row                             '获得选择单元格的行号
04      For c = 2 To 5                                '遍历第 2 至 5 列的单元格
05          If Cells(q, c) = "" Then                  '如果单元格为空
06              GoTo cuo                              '跳转到出错程序
07          End If
08          p = Cells(q, c) + p                       '非空则成绩累加
09      Next
10      m = p / 4                                     '计算平均分
11      Cells(q, 6) = m                               '平均分写入单元格
12      GoTo tuich                                    '退出过程
13  cuo:                                              '出错行标签
14      MsgBox "有空单元格，无法计算平均分！"            '显示出错信息
15  tuich:                                            '退出行标签
16  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，如果选择单元格所在行的成绩没有空白，则计算平均分，如图 6.40 所示。否则显示提示，并退出过程，如图 6.41 所示。

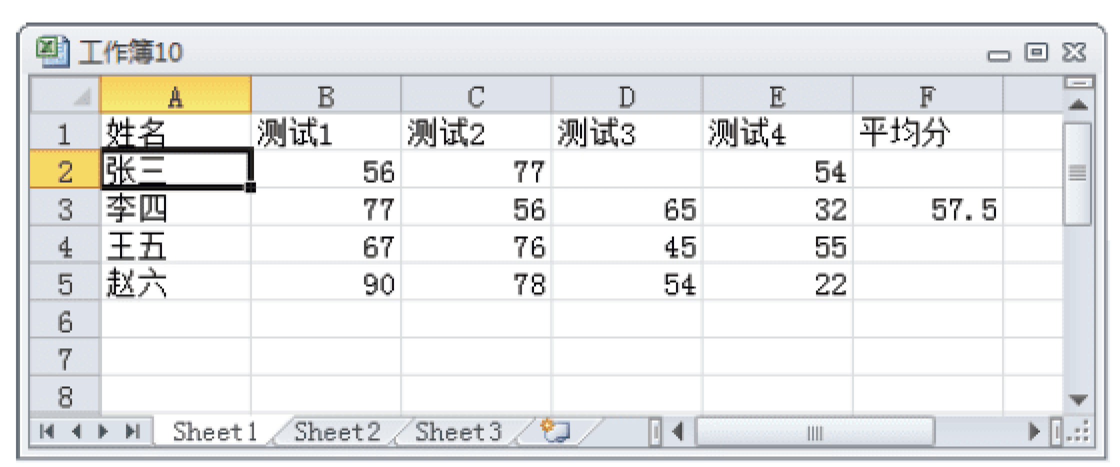


图 6.40　计算选择单元格所在行的平均分

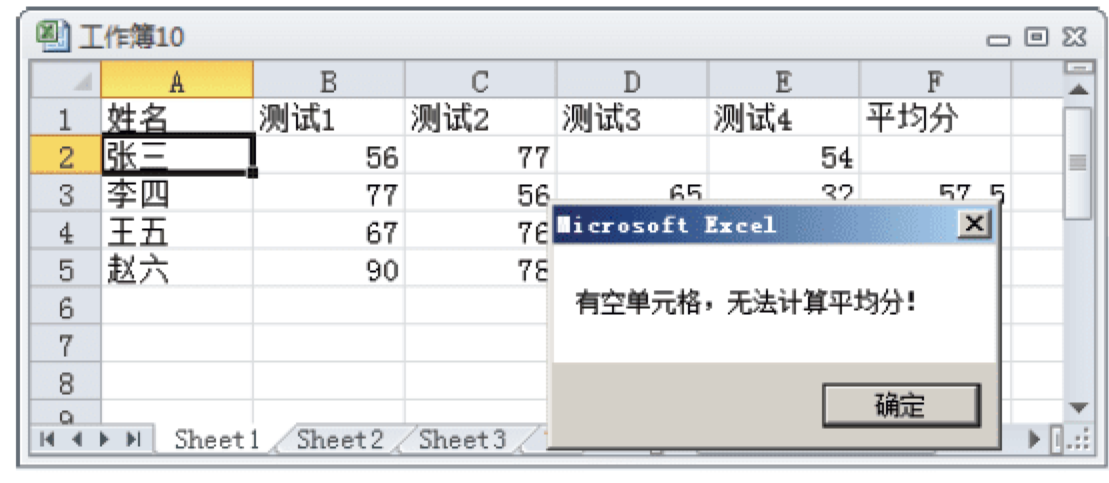


图 6.41　有空白行给出提示信息

【代码解析】本段程序代码展示了 GoTo 语句的用法，程序中以行标签来定位程序跳转的位置。在程序代码的第 06 行判断分母是否为 0，如果为 0 则使用 GoTo 语句将程序直接跳到行标签为“cuo”的语句，从而退出循环结构。如果正确完成计算，则在第 11 行使用 GoTo 语句跳转到行标签为“退出”的语句退出程序。该 GoTo 语句在程序中必须存在，否则在正常完成计算后，程序同样会执行其下的第 13 行代码，仍然显示出错休息。

注意：GoTo 语句虽然是很简单的一个语句，其功能却很灵活。GoTo 语句可以很方便地跳转到其他语句行，但给程序的结构化设计会带来一定的麻烦。如果在大型复杂的程序中频繁地使用 GoTo 语句，程序有可能会崩溃。除非必要，建议读者尽量少地使用 GoTo 语句，尽可能用其他循环结构来替代。

6.4　异常处理语句

无论程序员在设计时多么认真仔细，程序在编写过程中的错误是在所难免的，这就需要程序必须具有能够准确而快速地处理错误的能力。在程序设计过程中，有些错误是可以预见的，如分母为 0、输入值超过了取值范围等。在 VBA 中，可以利用错误捕获程序来捕获这些错误，并对错误进行适当的处理。这样设计出来的程序将具有更强的适应性。

6.4.1　On Error 语句

在 VBA 中，根据错误的性质的不同，VBA 的错误一般分为语法错误、运行时错误和逻辑错误 3 类。在 VBA 中处理错误一般需要设置错误捕获、编写错误处理程序和返回正常程序这 3 步。

在 VBA 中使用 On Error 语句来捕获错误，激活错误处理程序。该语句告诉 VBA 当程序运行时如果发生了错误该怎么办。On Error 语句通常有 3 种使用形式，如果在程序发生错误时，需要程序跳到 line 标号处执行错误处理程序，可以使用下面的语句：

```
On Error GoTo line:
```

警告：这里，标号 line 必须位于与 On Error 语句相同的过程中，否则将会产生编译时间错误。

当运行错误时，需要程序跳转到发生错误语句的下一条语句继续执行程序，可以使用下面的语句。

```
On Error Resume Next:
```

注意：这个语句将使程序对错误不产生反应，程序继续执行。

如果需要程序停止在当前过程中处理错误，可以使用下面的语句。

```
On Error GoTo 0
```

注意：使用上面语句时，即使程序中存在标号为 0 的语句，程序也不会将该语句作为错误处理程序的起点。

【范例 6-16】 计算用户输入数字的平方根，如果输入负数，提示错误，代码如下所示。

```
Sub 计算输入数的平方根()
    Dim a As Double, b As Double                    '声明变量
    Do While True                                   '开始循环
        On Error GoTo check1                        '捕获第一个可能的错误
        a =InputBox("输入数字计算平方根", "平方根计算器")
                                                    '显示数据输入对话框
        On Error GoTo check2                        '捕获第二个可能的错误
        b = Sqr(a)                                  '计算平方根
```

```
08          MsgBox "输入数据平方根为：" & b, vbOKOnly, "计算结果" '显示结果
09        Loop
10   check1:                                          '错误处理语句 1 行标签
11      MsgBox "输入数据非法，程序将退出", vbOKOnly, "提示"  '错误提示
12      Exit Sub                                        '退出过程
13   check2:                                            '错误处理语句 2 行标签
14      MsgBox "输入负值，无法计算平方根，程序自动退出", vbOKOnly, "提示"
                                                        '提示错误
15   End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序打开输入对话框，用户输入需要求平方根的数字，如图 6.42 所示。单击“确定”按钮关闭该对话框后，得到输入数字的平方根，如图 6.43 所示。





图 6.42　显示输入对话框可输入数字



图 6.43　显示计算结果

如果输入负数，给出提示对话框，如图 6.44 所示。单击对话框的“确定”按钮，程序退出。如果输入的非数字，则程序同样会给出提示对话框，如图 6.45 所示。



图 6.44　输入负数时的出错提示



图 6.45　输入非数字时的提示

【代码解析】本段程序演示程序中错误捕获和处理的方法。在这段程序中有两个可能出错的地方，一个就是第 05 行。当用户输入数据时，可能输入的不是数字，此时由于变量 a 定义为 Double 型，程序就会出错。因此在这里放置了一个 On Error GoTo 语句，该语句使程序跳转到指定的行标签处执行错误处理程序，显示出错提示并退出过程。第二个可能出错的地方就是当用户输入负数时，无法计算平方根。因此在第 07 行前同样放置了一个 On Error GoTo 语句，以便输入负数时给出提示。

提示：一般情况下，捕获错误的语句放置于程序可能出错的位置，也可以直接放置在过程的开始位置。程序运行时，行标签所指的程序行是错误处理的开始行。

6.4.2　Resume 语句

在进行错误处理时，当错误处理程序运行结束后，往往需要继续下面程序的执行，此时可使用 Resume 语句来让程序继续运行下去，其语法如下所示。

```
Resume
```

这里，当错误和错误处理程序处于同一个过程中时，从产生错误的语句处开始恢复程序运行。如果错误和错误处理程序不在同一个过程中，则从最后一次调用包含错误处理程序过程的语句处恢复程序运行。

如果错误和错误处理程序出现在同一个过程中时，需要程序从紧随产生错误的语句的下个语句恢复运行，可以使用下面语句：

```
Resume Next
```

注意：使用上面语句时，如果错误发生在调用的过程中，则程序找出最后一次调用包含错误处理程序的过程的语句，从紧随该语句之后的语句处恢复运行。

如果需要错误处理程序结束后，跳转到行标签为 line 处恢复程序运行，可以使用下面语句：

```
Resume line
```

注意：这里的行标签 line 必须和错误处理程序在同一个过程中。

【范例 6-17】 对范例 6-16 进行修改，在错误提示对话框中添加“确定”按钮和“取消”按钮。单击“确定”按钮运行用户重新输入的新数据，单击“取消”按钮，程序退出，代码如下所示。

```
Sub 计算输入数的平方根()
    Dim a As Double, b As Double                    '声明变量
start:
    a = 0
    Do While True                                   '开始循环
        On Error GoTo check1                        '捕获第一个可能的错误
        a = InputBox("输入数字计算平方根", "平方根计算器")'显示数据输入对话框
        On Error GoTo check2                        '捕获第二个可能的错误
        b = Sqr(a)                                  '计算平方根
        MsgBox "输入数据平方根为：" & b, vbOKOnly, "计算结果"  '显示结果
    Loop
check1:                                             '错误处理语句 1 行标签
    r = MsgBox("输入数据非法，是否继续？", vbOKCancel, "提示") '错误提示
    If r = vbOK Then Resume                        '单击确定按钮再次输入
    Exit Sub                                        '退出过程
check2:                                             '行标签
    r = MsgBox("输入数据为负数，是否继续？", vbOKCancel, "提示")'错误提示
    If r = vbOK Then Resume start                   '跳转到“start”处
    If r = vbCancle Then Exit Sub                   '退出过程
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序打开输入对话框。与范例 6-16 相比，当输入非数字时，给出提示对话框，如图 6.46 所示。单击“确定”按钮将重新打开输入对话框，单击“取消”按钮将退出程序。当输入负数时，程序给出提示，如图 6.47 所示。单价“确定”按钮后将退出过程。

【代码解析】与范例 6-16 相比，为了实现功能，有下面这些改动。在这里，两个提示对话框的样式都发生了改变，都增加了“取消”按钮。如果输入非数字数据，程序提示输入数据非法。此时单击提示对话框的“确定”按钮，MsgBox 函数返回值为 vbOkOnly，使

用 Resume 使程序继续执行出错语句，否则将退出程序。当输入值为负数，程序提示输入了负数，如果单击了提示对话框的“确定”按钮，第 19 行代码能够使程序从行标签“start”处重新开始。

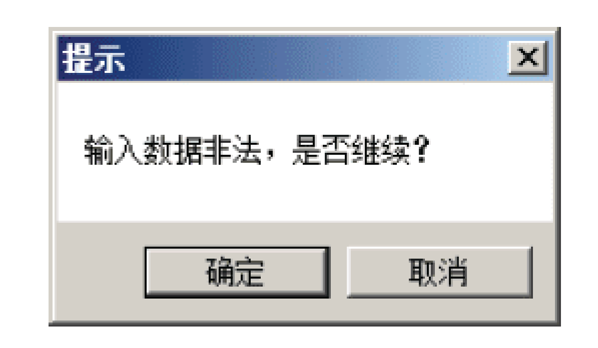


图 6.46　输入非数字时的提示对话框

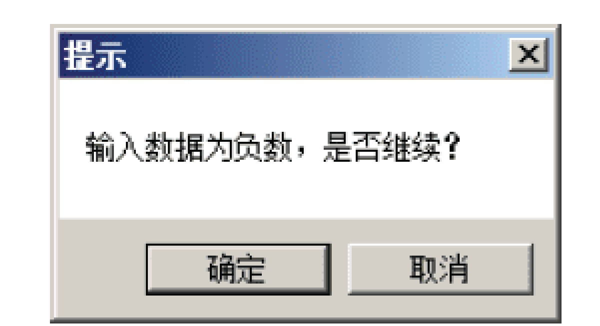


图 6.47　输入负数时的提示对话框

在“check2”错误处理程序中，如果使用 Resume 语句，则程序将会重新执行出现错误的第 09 行代码，而此时变量的值又没有更新，则第 09 行执行时仍然错误，此时将会进入一个死循环。因此，在本范例中的“check2”错误处理程序是从行标签“start”处开始，并且添加了赋值语句 a=0 将变量初始化。

注意： 在进行错误处理时，如果错误处理程序能够改正错误，则可以使用 Resume 语句。如果不能改正错误而使用了该语句，则错误会一直存在，这就构成了一个死循环。为了避免这种情况，有时也可以使用 Resume Next 语句跳过出错语句。

6.5　小　　结

本章学习了 VBA 程序中的控制结构，控制结构语句用于实现对程序流向的控制，包括选择结构和循环结构。使用选择结构能够对给定的条件进行分析和判断，根据条件的不同决定不同的操作，条件结构的使用使程序具有了智能化。控制结构的使用，是为了充分发挥计算机计算速度快的优势，使用重复的动作来完成某项任务。这两种结构是程序设计的基础，只有掌握它们才能获得真正意义的上应用程序。

从本章开始，读者将要接触到复杂应用程序的设计。应用程序的设计，应首先对问题进行分析，明确需要解决的问题，确定问题解决的方案和步骤。这里问题解决方案和步骤的问题实际上就是编程中常说的算法。确定的算法应该注意要具有穷性（即在执行有限步后能停止）、确切性、保证输入和输出，同时还应该具有可行性。当应用程序比较复杂时，可以使用流程图来使算法清晰、直观和形象，为最终程序的设计打下良好的基础。

通过本章的学习，相信读者已经掌握了 VBA 的基本程序结构，第 7 章将开始学习 VBA 中模块、过程和函数的知识，对 VBA 进行更深入的探究。

6.6　本 章 习 题

1．图 6.48 所示的流程线结构反映的是下面哪种选择结构？（　　）

A．Select Case　　B．If…Then…ElseIf　　C．If…Then…Else　　D．If…Then

2．图 6.49 所示的流程线结构反映的是哪种循环结构？（　　）

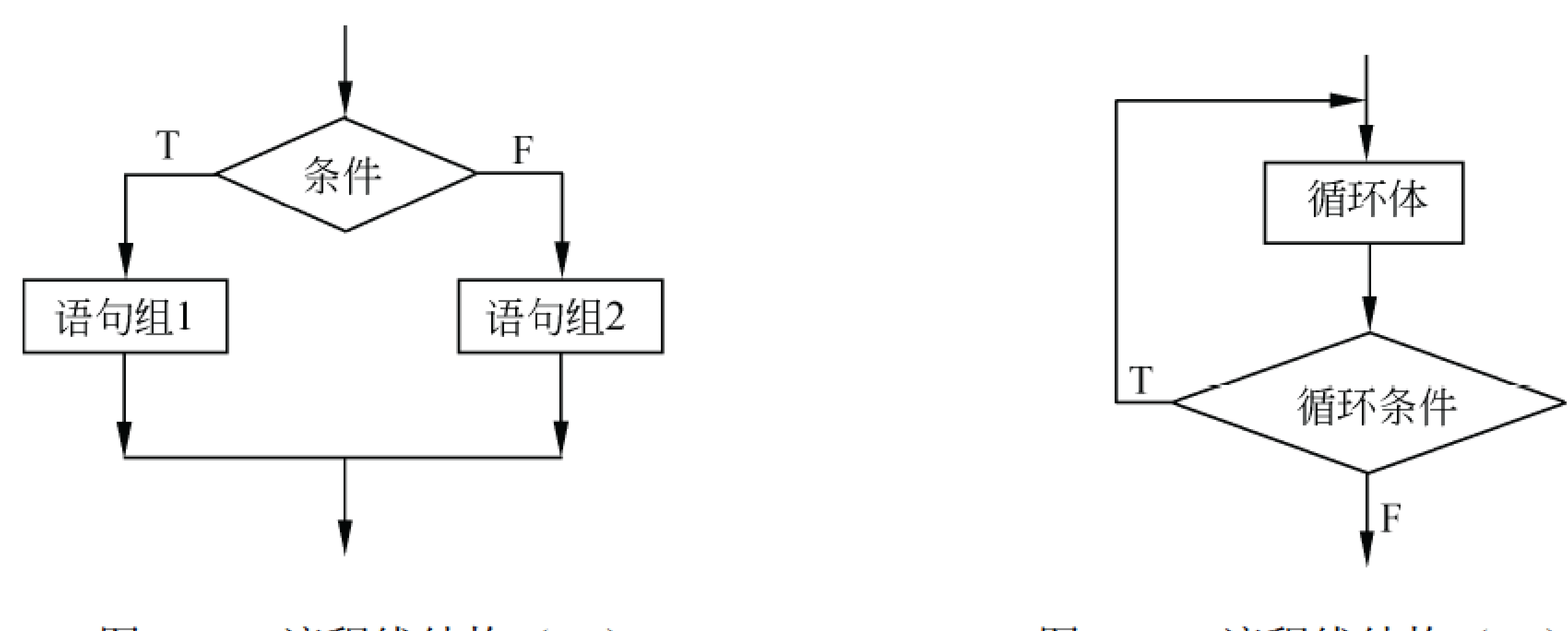


图 6.48　流程线结构（一）　　图 6.49　流程线结构（二）

A．For…Next　B．Do...Loop While　C．Do While...Loop　D. While…Wend

3．结束 Sub 过程代码的运行应该使用下面的哪个语句？（　　）

A．Exit Property　B．End　C．Exit Sub　D．Stop Sub

4．选择下面程序的运行结果。（　　）

```
Sub test ()
Dim n As Integer
Do
      n=n+1
      Debug.Print n
Loop While n<0
End Sub
```

A．0　B．1

C．2　D．没有任何输出结果

5．VBA 中最基本的 3 种程序结构是什么？

6．常用的选择结构语句有哪两个？

7．常用的循环结构语句有哪些？

8．编写一个程序，求出所有的“水仙花数”。“水仙花数”是指一个 3 位的整数，其各位数字立方和等于给数本身。例如，153 即是水仙花数，因为 $153=1^3+5^3+3^3$。

【提示】可以用循环结构来遍历所有 3 位整数，通过给出的公式来确定“水仙花数”。

9．编写一个程序，求解百钱买百鸡的问题。雄鸡 7 元一只，母鸡 5 元一只，小鸡 1 元 3 只，问花 100 元买 100 只鸡，但要求雄鸡、母鸡和小鸡都必须买，则雄鸡、母鸡和小鸡应该各买几只？

【提示】该首先确定算法。设 x、y、z 分别为雄鸡、母鸡和小鸡的个数，则可以得到以下的方程：

$$5x+3y+z/3=100$$

$$x+y+z=100$$

即雄鸡、母鸡和小鸡的个数必须同时满足上面两个方程。同时，还要注意，这里雄鸡、母鸡和小鸡的个数都是小于 100 的。此时，可以可用循环控制来遍历符合条件的数，这些数就是这个题目的解。

第 7 章　使用 VBA 数组

在数据类型一章，将具有相同性质的数据归为一种类型，只是在理解上进行了分类。如果要对同一类型的多个数据进行处理，就要定义多个变量，对其进行存储和运算，例如有 30 个学生的成绩需要处理，则需要申请 30 个变量，此种做法显然是不合实际的。数组可以解决此类问题，其能够存储同一类型的多个变量。本节主要学习的内容有：

- ❑ 理解数组的基本概念；
- ❑ 掌握一维数组、二维数组的声明、初始化以及数组的赋值；
- ❑ 掌握静态数组的初始化、赋值方法；
- ❑ 掌握动态数组的声明、方法，了解复制数组和清除数组的方法。

7.1　什么是数组

数组是一种特殊类型的变量，它是一组变量的集合，变量具有相同的数据类型，同时共享同一个名字。对于大量的有序的数据，可以使用数组来对其进行存储和处理。在 VBA 中，数组中各元素可以是相同的数据类型，也可以是不同的数据类型。

7.1.1　数组的概念

数组是一种特殊类型的变量，它是一组变量的集合，变量具有相同数据类型，同时共享同一个名字。对于大量的有序的数据，可以使用数组来对其进行存储和处理。在 VBA 中，数组中各元素可以是相同的数据类型，也可以是不同的数据类型。数组用来保存一组有序的数据，具体的形式如图 7.1 所示。

这样的数据，可以通过在项目名称后加上序号表示出来，如，数据“郭刚”，可以通过“项目（1）”来表示。这种只具有行或者列的数据、可以用单独的序号表示的数组，称为一维数组。数据在数组中的序号，称为下标。下标的最大值称为下标上界，下标的最小值称为下标下界。

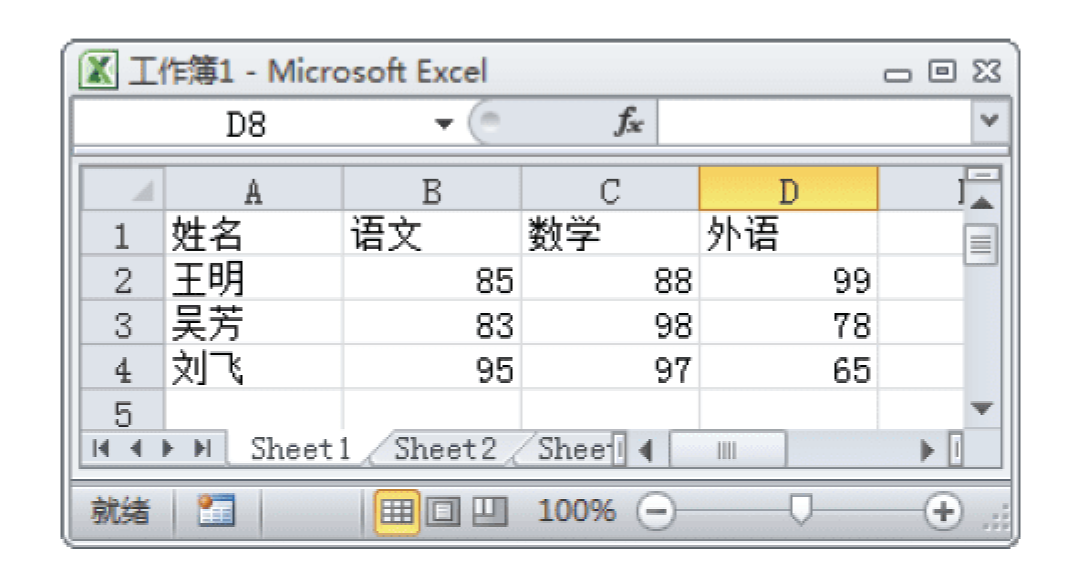
如果将由行和列组成的数据表定义为一个数组，这个数组就是二维数组。Excel 工作表实际上就可以看成是一个二维数组，表中元素的位置由行号和列号来确定，如图 7.2 所示。

提示： 对于数组来说，如果除了行和列上的排列外，还具有深度，就像 Excel 工作簿中的多个工作表，那么这样结构的数组就是三维数组。VBA 的数组最大能够达到 60 维，但是超过了三维的数组无论是在理解上还是在操作上都是很难的，因此多维数组在编程时很少用到。

项目（1）	郭刚
项目（2）	男
项目（3）	大学本科
项目（4）	武汉市



图 7.1　数组示意



	A	B	C	D
1	姓名	语文	数学	外语
2	王明	85	88	99
3	吴芳	83	98	78
4	刘飞	95	97	65
5				

图 7.2　二维数据表

7.1.2　声明一维数组

在程序中使用数组的时候，需要先声明数组，也就是定义数组。数组的声明和变量的声明是一样的，使用 Dim、Static、Private 和 Public 语句来实现。在进行数组声明时，需要指定数组的大小。如果数组大小是固定的，这种数组称为固定大小的数组，即静态数组。如果程序运行时，数组的大小可以改变，则数组就是一个动态数组。对于固定大小的一维数组，可以有 2 种方法来定义。第 1 种方法的语法格式如下：

```
Dim 数组名 (上界) As 数据类型
```

参数说明如下所示。

- ❑ 数组名：数组的名称。
- ❑ 上界：数组下标的上界值。
- ❑ 数据类型：数组变量的数据类型。

该方法只给出数组下标的上界，即可以使用的最大序号值，省略了数组下标的下界。此时下界的默认值为 0，即数组中元素的个数是从 0 开始到定义上界。例如：

```
Dim MyArray (20) As Integer
```

这里定义了一个名为 MyArray 的一维数组，数组中的元素从 MyArray(0)开始，依次是 MyArray(1)、MyArray(2)…… MyArray(20)，数组一共包含 21 个元素。

提示： 数组下界的默认值可以在编写程序时自定义为 0 或 1。但这种修改下界默认值的语句只能出现在用户窗体或者模块的声明部分，不能出现在过程中，且该语句必须放在数组定义之前。如果将下界默认值设置为 1，可以使用下面的语句：

```
Option Base 1
```

定义数组的第 2 种方法，是在定义数组时，指定数组的下界和上界的值。此时，数组的下界值不是从默认的 0 开始。这种方法的语法格式如下：

```
Dim 数组名 (下界 To 上界 ) As 数据类型
```

参数说明如下所示。

- ❑ 数组名：数组的名称。

- ❑ 上界 To 下界：二维数组下标的下界和上界的值。
- ❑ 数据类型：数组变量的数据类型。

【范例 7-1】 定义一个有 12 个元素的一维数组，数组起始元素的下标的下界为-2，代码如下所示。

```
01   Sub 定义一维数组()
02       Dim MyArray(-2 To 9)                    '定义一个数组变量
03       Debug.Print  UBound(MyArray)            '显示数组的最大下标
04       Debug.Print  LBound(MyArray)            '显示数组的最小下标
05   End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中将显示定义的一维数组的最大下标（即上界）和最小下标（即下界）。程序运行的结果如图 7.3 所示。

【代码解析】示例展示了一维数组的定义方式。在这段代码的第 02 行中定义了一个一维数组，数组下标的下界是-2，上界是 9。数组中的对应元素是 MyArray（-2）、MyArray（-1）…… MyArray（9）。代码第 03 行使用的 UBound 函数是 VBA 的数组函数，用于获得数组下标的上界。在代码中使用的 Print 语句，可在调试程序时在“立即窗口”中显示指定的内容。



图 7.3　程序运行的结果

🔔**注意：在定义数组时必须注意下面几点：**

- ❑ 与变量命名相同，数组的命名应该符合标识符的特点，并且具有一定的意义，这样便于程序阅读。
- ❑ 在同一个过程中，数组名不能与已存在的变量名相同。
- ❑ 在定义数组，上界和下界只能使用常数而不能使用变量，否则就会出错。

7.1.3　声明二维数组

二维数组的定义方式与一维数组类似，只是需要设置两个下标的上界和下界。在定义二维数组时，可以像定义一维数组那样省略下标的下界，其语法格式如下：

```
Dim 数组名 (第一维上界，第二维上界) As 数据类型
```

参数说明如下所示。

- ❑ 数组名：数组的名称。
- ❑ 第一维上界：定义数组第一维的下标上界。
- ❑ 第二维上界：定义数组第二维的下标上界。
- ❑ 数据类型：数组变量的数据类型。

在定义二维数组时，也可以完整地定义数组二维的下标的上界和下界，其语法格式如下：

```
Dim 数组名 (第一维下界 To 第一维上界，第二维下界 To 第二维上界 ) As 数据类型
```

参数说明如下所示。

- ❑ 数组名：数组的名称。

- 第一维下界 To 第一维上界：定义数组第一维下标的上界和下界。
- 第二维下界 To 第二维上界：定义数组第二维下标的上界和下界。
- 数据类型：数组变量的数据类型。

【范例 7-2】 定义一个有 12 个元素的二维数组，数组起始元素的下标为-2，代码如下所示。

```
01  Sub 定义二维数组()
02      Dim MyArray(-2 To 9,8 To 12)                '定义一个数组变量
03      Debug.Print  UBound(MyArray,1)              '显示第一维下标上界
04      Debug.Print  UBound(MyArray,2)              '显示第二维下标上界
05      Debug.Print  LBound(MyArray,1)              '显示第一维下标下界
06      Debug.Print  LBound(MyArray,2)              '显示第二维下标下界
07   End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中将依次显示二维数组二维的下标的上界和下界的数值。程序运行的结果如图 7.4 所示。

【代码解析】示例用于展示二维数组的定义方式。代码的第 03 行使用 UBound(MyArray,1)语句获得二维数组第一维下标的上界，其中参数 1 是数组的维数值。第 04 行代码至第 06 行代码分别获得数组第二维下标上界的值以及二维数组两个维的下标下界值，并将它们依次显示在“立即窗口”中。

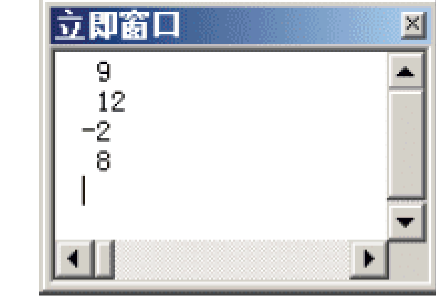


图 7.4　程序运行的结果

注意：在 VBA 中，定义多维数组的方式和二维数组的定义方式一致。但要注意的是，定义了大数据量的数组将会占用极大的内存空间。在定义多维数组时应该考虑到系统的配置及程序运行速度，尽量避免使用数据量大的数组。

7.2　静 态 数 组

在学习了如何定义数组之后，就可以在程序中先定义数组变量，然后在程序中使用数组变量，本节将介绍静态数组的使用方法。

7.2.1　初始化静态数组

在 VBA 中，创建数组后将自动初始化数组中的每个元素。对于数值型数组，数组中每个元素将自动清零。对于字符串型的数组，数组中的每个元素将自动初始化为空的字符串。但在对数组中的元素进行处理时，仍需要对其进行初始化。数组的初始化，实际上是对数组的赋初值的过程，即向数组中输入需要的数据。通过下面的示例将展示数组初始化的基本方法。

【范例 7-3】 使用 Array()函数给数组赋值的方法，如下所示。

```
Sub ArrayInit()
    Dim MyArray,a As Integer                          '定义数组和变量
    MyArray=Array("一月","二月","三月","四月","五月","六月")
```

```
                                    '使用 Array 函数给数组赋值
04      For a=0 To 5
05          Debug.Print  MyArray(a)           '在立即窗口中显示数组值
06      Next
07   End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中出现程序运行结果，如图 7.5 所示。

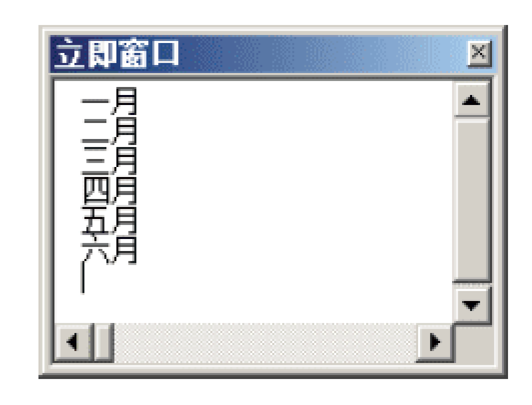


图 7.5　顺序结构演示结果

【代码解析】示例展示了使用 Arry 函数给数组赋值的方法。在这段代码中，第 03 行代码中使用 Array()函数将字符串“一月”、“二月”…… 分别赋值给数组 MyArray 的各个对应元素。在第 04～06 行代码中，使用 For…Next 循环遍历数组中的每个元素，将它们依次显示在“立即窗口”中以显示赋值的结果。

警告： Array()函数返回值是 Variant 类型，在使用该函数为数组赋值时，数组不能设置下标，否则将会出现“不能给数组赋值”的错误提示。

7.2.2　使用二维静态数组

二维数组的应用最容易想到的就是矩阵，利用二维数组可以完成矩阵的加、减、乘、除运算。二维数组在使用时，首先要声明静态二维数组，声明时必须指明数组的维数和数组的上下界，上下界的值必须为常数，处理二维静态数组，常采用嵌套循环。二维数组的处理如例 7-4 所示。

【范例 7-4】 此例中定义了一个二维数组 a，然后利用 For...Next 语句将其值初始化。然后将其值输出到立即窗口中。

```
01  Sub 矩阵赋值()
02    Dim i As Integer, j As Integer
03    Dim a(1 To 30, 1 To 5) As Integer
04        '用值填入数组
05    For i = 1 To 30
06       For j = 1 To 5
07          a(i, j) = i * j
08       Next j
09    Next i
10    For i = 1 To 30
11      For j = 1 To 5
12        Debug.Print "a(" & i & "," & j & ")" & " = " & a(i, j) & Space(5);
13      Next j
14      Debug.Print
15    Next i
16  End Sub
```

【代码解析】第 2～3 行用于声明整型变量，第 4～8 行为数组赋值，第 9～14 行用于输出各行的值。执行结果如图 7.6 所示。

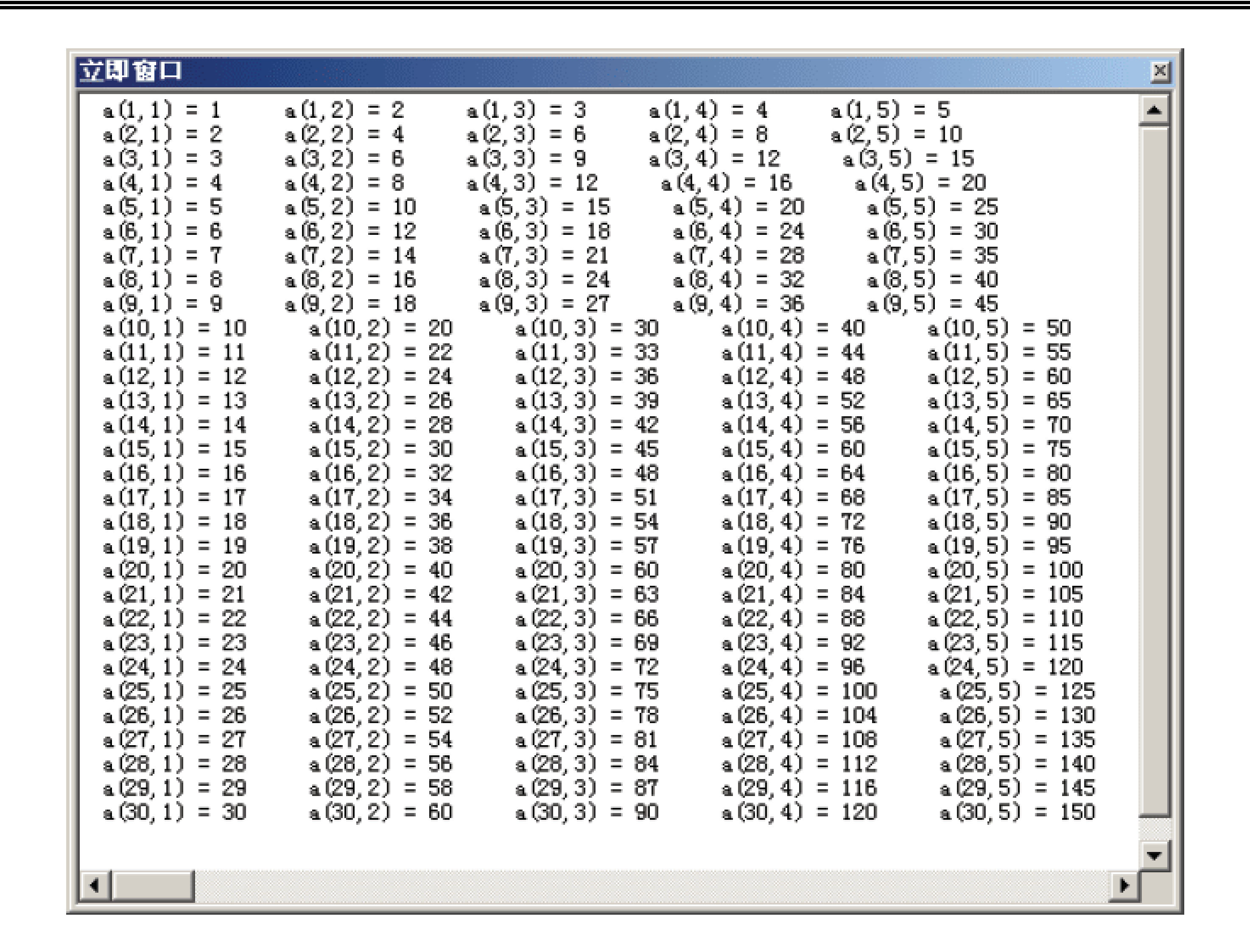


图 7.6　数组赋值

7.3　动 态 数 组

动态数组是指在程序运行时大小可以改变的数组。在定义静态数组时，下标的上界和下界只能使用常量来设置。而对于动态数组来说，则可以使用变量来设置下标的下界。

7.3.1　声明动态数组

使用动态数组可以根据用户需要调整数组的大小，有效利用存储空间；当数组使用完毕后，可以将内存释放给系统。声明动态数组时，不要指定数组的上下界和数组的维数，将“()”中的内容置空。其余语法同于静态数组的声明。其语法描述如下所示。

```
Dim 数组名() As 数据类型
```

声明时还可以使用 static、public、private 等关键字，在代码的例子中声明了一个 Double 型的动态数组。

```
Dim a()As Double
```

注意：可以在过程中使用 ReDim 语句来做隐含性的数组声明。当使用 ReDim 语句时要小心点，不要拼错数组的名称。否则即使在模块中有包含 Option Explicit 语句，仍然会因此而生成第 2 个数组。

7.3.2　定义数组大小

在声明了动态数组之后，就会在需要的地方定义数组的大小。在程序代码中可以使用 ReDim 语句去定义其维数，去定义元素的数目以及每个维数的底层绑定。每当需要改变数

组的上下界和维数时，也可以使用 ReDim 语句。在使用 ReDim 语句改变数组大小时，其下标可以是常量，也可以是数值已经确定的变量。在更改数组大小时，可以不指定数组的类型，默认为当 Dim 语句中声明的数据类型相同。如果在 ReDim 语句中指明了数据类型，也必须与 Dim 语句中声明的数据类型一致。其语法描述如下所示。

```
ReDim 数组名（[数组的下界 To ] 数组的上界，……）
```

例如重定义数组 a 的大小为 10 如下述代码所示。

```
Dim a() As Integer
ReDim a(1 To 10)
```

对于已经存在元素的数组，改变其大小和维数时，对其重新分配存储空间，会丢失原有数组中的元素。此时就需要为 ReDim 语句加上 Preserve 关键字。其语法描述如下。

```
ReDim Preserve 数组名（UBound(数组名)+n）
```

此语句仅能改变多维数中最后一维的上界，如果改变了其他维数或最后一维的下界，程序运行时就会出错。

注意：ReDim 只能被用来改变数组中元素的数目，如果试图用 ReDim 改变数组的数据类型，将会导致程序出错。

例如下述代码就将原有数组的扩大，数组中的元素增加 5 个，并且不丢失现有的元素。

```
ReDim Preserve b(UBound(b)+5)
```

【范例 7-5】 声明一个动态数组，在过程中重新定义数组的大小，使每次运行程序数组的上界都是一个随机值，代码如下所示。

```
01    Dim MyArray() As Integer                    '声明数组
02    Sub 动态数组()
03       Dim i As Integer                         '定义变量
04       i = Rnd * 10                             '生成随机数
05       ReDim Preserve MyArray(i)                '重新声明数组
06       Debug.Print  UBound(MyArray)             '显示数组的上界
07    End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中将显示随机产生的数组下标上界的值。多次按 F5 键，将显示一系列不同数组上界值。多次运行程序的结果如图 7.7 所示。



图 7.7　程序运行的结果

【代码解析】示例代码将产生随机大小的动态数组，每次运行程序，数组大小都会不同。在代码中，第 01 行代码首先声明一个数组，这个数组在声明时括号内必须为空。Rnd 函数可以产生参数一个小于 1 且大于等于 0 的随机数，第 03 行代码将 Rnd 函数产生随机数乘以 10 得到整数随机数。代码的第 05 行对数组进行重新声明以更改数组的大小。

7.3.3　复制数组

在定义动态数组时，首先应在用户窗体、模块或过程中使用 Dim 或 Public 声明一个没

有下标的数组，然后再在过程中用 ReDim 语句来重新定义数组的大小。

数组中的元素，可以像普通变量那样进行访问和赋值，因此可以实现将一个数组中的值复制到另一个数组中。

【范例 7-6】 复制数组元素的方法如下所示。

```
01  Sub 数组元素的复制()
02      Dim MyArray1(5) As Integer, MyArray2 As Variant    '定义两个数组
03      Dim a As Integer                         '定义变量
04      For a = 0 To 4
05          MyArray1(a) = a                      '给数组 MyArray1 赋值
08      Next
09      MyArray2 = MyArray1                      '复制 MyArray1 元素到 MyArray2
10      For a = 0 To 4
11         Debug.Print  MyArray2(a)              '显示数组 MyArray2 元素
14     Next
15   End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中显示数组中每个元素的值。程序运行的结果如图 7.8 所示。

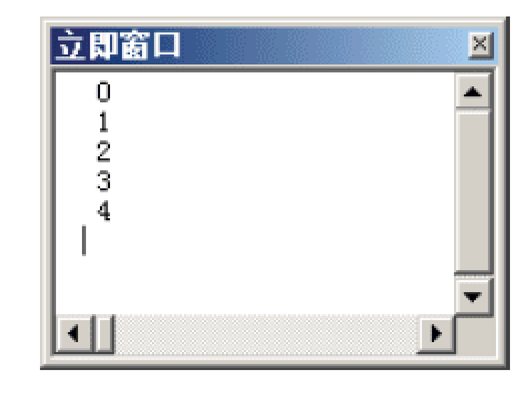


图 7.8　程序运行的结果

【代码解析】本示例展示对数组进行复制的过程。在本示例中，在代码的第 02 行定义两个数组。第 04～08 行使用 For…Next 结构将当前变量 a 的值赋予数组 MyArray1。代码第 09 行对数组 MyArray2 赋值，赋值的过程即是数组复制的过程。

提示： 这里将数组 MyArray2 声明为一个 Variant 变量，则只需要通过 MyArray2 = MyArray1 语句直接赋值即可实现数组数据的复制。如果不定义为 Variant 变量，则需要使用 For…Next 语句依次给 MyArray2 中的每个元素进行赋值才能实现数组元素的复制。

7.3.4　清空数组或重定义数组

在使用数组时，有时需要清除数组的内容或对数组重新定义。此时可以使用 Erase 语句来完成，其语法结果如下：

```
Erase 数组名 1,数组名 2 ……
```

参数表示需要清除的数组名。

【范例 7-7】 清除静态数组 MyArray1 和动态数组 MyArray2 的方法如下所示。

```
Sub 清除数组()
    Dim MyArray1 (), MyArray2(8) As Integer      '定义两个数组
    Dim a As Integer                             '定义一个变量
    a=8                                          '变量赋值
    ReDim MyArray1(a)                            '重定义动态数组
    For a = 0 To 8
        MyArray1(a)= a                           '给 MyArray1 数组赋值
        MyArray2(a)=a                            '给 MyArray2 数组赋值
    Next
    Erase MyArray1,MyArray2                      '清除两个数组
```

```
11        For a=0 To 8
12            Debug.Print  MyArray2(a)          '显示数组 MyArray2 的值
13        Next
14        Debug.Print  MyArray1(1)              '显示数组 MyArray1 第一个元素的值
15    End Sub
```

【运行结果】按 F5 键运行程序，在“立即窗口”中显示数组 MyArray2 各个元素的值，如图 7.9 所示。同时，程序将显示出错信息，如图 7.10 所示。



图 7.9　程序运行的结果

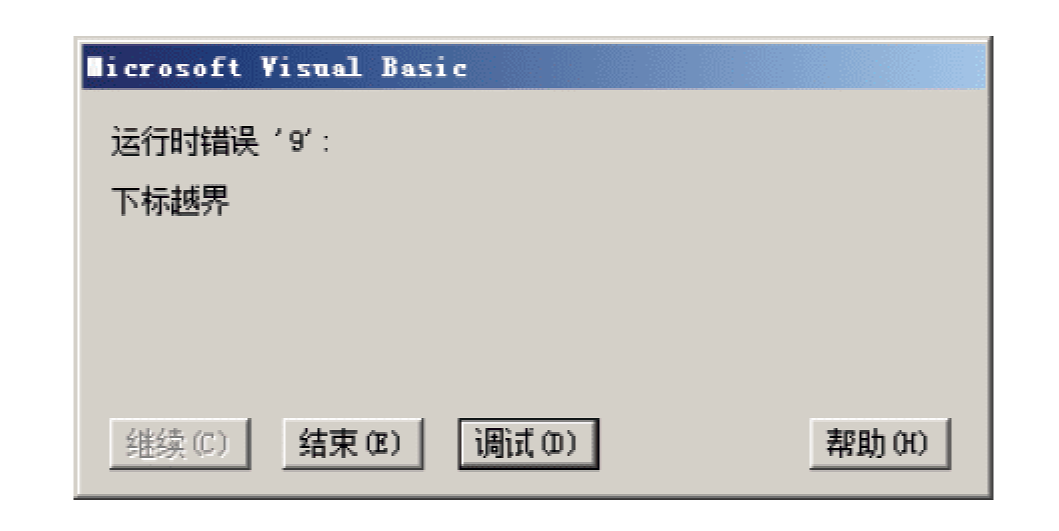


图 7.10　程序运行时的出错信息

【代码解析】本示例展示清除数组元素的方法。在本示例中，定义了两个数组元素，一个是动态数组 MyArray1，另一个是静态数组 MyArray2。在对数组赋值后，使用 Erase 语句删除这两个数组。这里要注意，在使用 Erase 语句时，两个数组值给出数组名即可，不能带括号和下标。对于动态数组来说，删除了数组，数组将不存在。在代码的第 13 行打印动态数组 MyArray1 的第一个元素，将得到错误提示。

提示： Erase 语句删除数组时，将根据数组的不同类型有不同的行为。对于静态数组来说，根据数组数据类型的不同，删除后的数组内容也不同。如对于固定字符串数组，数组中元素将设置为空字符串""。而对于 Variant 数组，每个元素将设置为 Empty。对于动态数组，Erase 语句将删除数组结构并释放数组占用的内存。

7.4　小　　结

本章学习了 VBA 中过程和自定义函数的创建和使用方法。过程是 VBA 程序的基本单元，应用程序实际上就是一个个实现功能的过程的集合。通过本章的学习，读者将能够掌握过程的创建和调用的方法，以及调用过程时过程间传递参数的方法。VBA 的函数包括 VBA 自带的函数和用户自定义函数。通过本章的学习，读者应该掌握自定义函数的创建以及使用 VBA 程序和在工作表中的不同调用方法，能够熟练应用函数来扩展 Excel 的功能。

Function 函数和 Sub 过程都属于 VBA 的通用过程，它们都是构成程序的基本单位，能够使用 Private 和 Public 等关键字来设定作用域，两者都可以接收参数，并且参数的设置是相同的。但两者也有着许多的不同，Sub 过程是不能返回值的，而 Function 函数是可以返回值。Sub 过程可以作为 Excel 的宏来调用，而 Function 函数则不会出现在“宏”列表中，也就是说不能单独运行。Sub 过程在程序中可以作为单独的语句来调用，而 Function 函数通常是作为表达式的一部分来使用。

7.5　本 章 习 题

1．已知如下程序请指出程序执行结束后，i =______ ，j =______ ， k =______ 。

```
Sub 求数组的维数()
    Dim i As Integer, j As Integer, k As Integer
    Dim a(1 To 30, 1 To 5, 1 To 10) As Integer
     '用值填入数组。
    For i = 1 To 30
        For j = 1 To 5
            For k = 1 To 10
            a(i, j, k) = i * j * k
            Next j
        Next j
    Next i
    For i = 1 To 30
        For j = 1 To 5
            Debug.Print "a(" & i & "," & j & ")" & " = " & a(i, j) & Space(5);
        Next j
        Debug.Print
    Next i
    i = LBound(a, 1)
    j = UBound(a, 1)
    k = Ubound(a, 3)
End Sub
```

2．声明一个静态一维数组 a，用于存放 10 个整数。

3．编写一个 VBA 程序，定义一个字符串数组，并将数组中各元素的值依次设置为“小白兔”、“小乌龟”、“小老虎”、“多啦 A 梦”、“米老鼠”，并将赋值后的数组输出到立即窗口。

4．编写一个使用户能够为一个有 10 个元素的数组赋值的程序。

【提示】可以使用 InputBox 语句来获取用户输入值，使用 For…Next 语句给数组中所有元素赋值。

第 8 章　使用过程与函数

过程与函数是 VBA 程序设计中的重要知识点。VBA 应用程序是由过程组成的，过程分为 Sub 过程和 Function 过程，VBA 过程应根据需要设计，然后，在程序中调用过程，实现在程序对数据的处理。根据程序的需要，可以将程序中的变量传递到过程中，经过过程处理后，再返回处理的结果。本章的主要学习内容和学习目的如下：

- ❑ 理解 VBA 中的模块与过程；
- ❑ 掌握 Sub 过程的创建和调用方法；
- ❑ 掌握过程与函数中参数的传递方法；
- ❑ 掌握系 Function 过程的调用方法。

8.1　什么是过程

在 VBA 中，一个重要的概念就是过程。过程是程序中最小的逻辑单元，为了简化程序设计，其可以用来扩充或增强 Visual Basic 构建。而在 VBA 中，任何程序代码都需要有存放之地，这个存放地就是模块。可以这样说，模块是过程的容器，过程是代码的集合。本节将对 VBA 的模块和过程进行介绍。

8.1.1　初识 VBA 模块

在 VBA 中，模块是作为一个单元保存在一起的 VBA 定义和过程的集合，通俗地说就是程序代码存放的地方。模块一般分为两类，即标准模块和类模块。标准模块用于存放全局变量变、公共函数的说明、函数以及自定义函数等，用户可以在该模块的工程中对它们进行调用。类模块用于存放用户字句创建对象的定义，其可以跨工程调用。

在前面章节中，实际上就已经接触到了模块。例如，在录制宏的时候，宏会自动创建模块。在编写应用程序时，编程者也可以自己创建模块，方法是：选择“插入”→“模块”命令，VBA 会在当前的工程中创建一个模块。也可以在“工程”窗口中任意一个项目上右击，在弹出的快捷菜单中选择“插入”→“模块”命令，即插入一个模块，如图 8.1 所示。

模块是代码存放的场所，程序代码可以在模块中编写并运行。使用模块来管理代码，可以通过顶部的通用声明来定义模块级别的变量，使模块内的不同过程都能共享这些变量。同时，使用模块能够对过程和变量设置一定的权限，使其只能在一定的范围（即过程）内使用。

🔔注意：在进行 VBA 编程时，还会涉及到工程。模块是过程的集合，那么工程实际上就是程序中模块的集合。一般情况下，一个模块可以包含任意数量的过程，一个工程可以包含任意多个的模块。

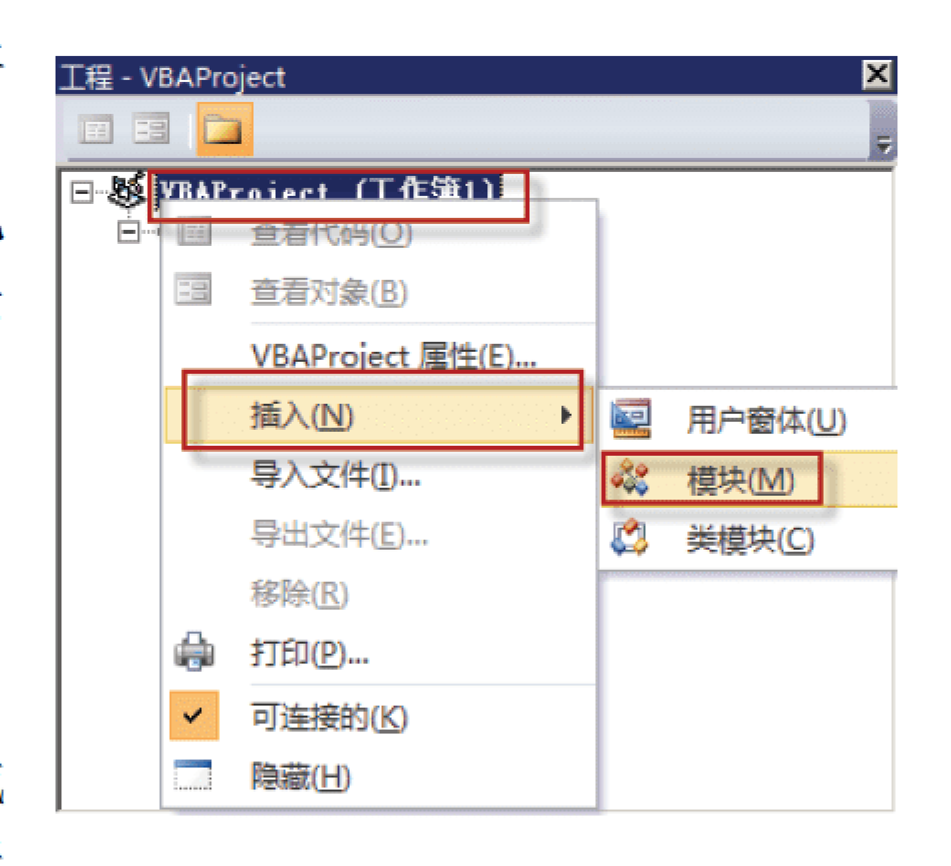


图 8.1　插入模块

8.1.2　理解过程

过程是一组能够完成特定任务的 VBA 语句所组成的代码的集合。在 VBA 中，所有的程序代码必须放置在过程中，通过启动过程来执行。过程可以分为事件过程、属性过程和通用过程这 3 类。

事件过程指的是当发生某个事件（如单击或打开工作簿等）时，对这个事件作出响应的程序段。如，当需要单击按钮开始动画，动画程序就可以放置于按钮的“Click”事件段中，如图 8.2 所示。



图 8.2　按钮的 Click 事件代码

在不同的事件过程中执行同一段相同的代码，这段代码在编程时可以独立出来作为一个单独的过程，这样的过程即为通用过程。在 VBA 中，通用过程包括 Sub（子程序）过程和 Function（函数）过程。

属性过程是专门为类模块编制的程序代码块。在创建类模块时，往往需要为类模块创建属性，这个属性可以通过编写属性过程来实现。

过程在使用时，必须要命名，程序以此来区分不同的过程，也只有在命名后，才能实现过程的调用。在 VBA 中，过程名可以是数字、字母、汉字和下划线，但不能包括空格、句号、感叹号以及@、#、$和&等符号。同时，过程名的第一个字母不能是数字或下划线，并且最多只能包括 255 个字符。

8.2　VBA 中的 Sub 过程

读者对 Sub 过程并不会觉得陌生，看看前面章节的示例，就会发现实际上这些示例都是一个或者简单或者复杂的 Sub 过程。Sub 过程是通用过程中的一类常见的过程，是 VBA 应用程序的基础。本节将介绍 Sub 过程的创建和调用。

8.2.1　创建 Sub 过程

Sub 过程的创建比较简单，在 VBA 中一般有两种方法。第一种方法是通过直接在模块中输入程序代码来创建 Sub 过程。Sub 过程的语法结构如下：

```
[Private|Public|Friend]|[Static] Sub 过程名 [(参数列表)]
    语句块
    [Exit Sub]
    语句块
End Sub
```

参数说明如下所示。

- ❑ Private：表示只有在包含其声明的模块中才能被访问，也就是该过程是一个局部过程。这种过程不能被其他模块中的过程访问。
- ❑ Public：表示所有的模块的所有其他过程都可以访问该过程，也就是该过程是一个公共过程。如果 Public 关键字省略，其默认为公用的。在 Excel VBA 中，各个工作表和用户窗体调用的公共过程都要在模块中使用 Public 关键字定义。
- ❑ Friend：只能在类模块中使用，表示该过程在整个过程中可见，但对对象实例的控制者却不可见。
- ❑ Static：表示在调用时保留 Sub 过程的局部变量值，也就是说过程中的局部变量都是静态变量。这些过程中的变量被调用后，其值仍将保留。但这里要注意，在 Sub 之外定义的变量不会受到影响，即使过程中使用了这些变量。
- ❑ End Sub：该语句表示 Sub 过程的结束，程序运行到此将结束当前的 Sub 过程的运行。为了能够正确运行，每个 Sub 过程必须有一个 End Sub 语句。
- ❑ Exit Sub：Sub 过程中允许有一个或多个 Exit Sub 语句，这些语句将使程序直接退出过程的执行。
- ❑ 语句块：是 Sub 过程中的可执行程序代码，这些程序代码实现 Sub 过程的功能。

注意： 在定义 Sub 过程时要注意，Sub 过程的定义是不能嵌套的，即不能在一个 Sub 过程中定义另一个 Sub 过程。

要创建 Sub 过程，只需要按照上面代码的语法规则在“代码”窗口中输入代码即可，就像前面章节的示例中那样。实际上，在 Visual Basic 编辑器中可以使用“添加过程”对话框来添加一个过程，具体的操作步骤如下：

（1）在工程资源管理器中插入一个模块，打开该模块的“代码”窗口，将输入点光标放置于“代码”窗口中。选择“插入”→“过程”命令，如图 8.3 所示。

（2）此时将打开“添加过程”对话框，在对话框的“名称”文本框中输入过程名，同时设置过程的类型和范围，如图 8.4 所示。

提示： 在“添加过程”对话框中，“子程序”单选按钮对应 Sub 过程，“函数”单选按钮即为 Function 过程，“属性”单选按钮为 Property 过程。选择“公共的”单选按钮时，过程代码中使用 Public 关键字。选择“私有的”选项，在过程代码中使

用 Private 关键字。选中“把所有局部变量声明为静态变量”复选框，过程名前添加 Static 关键字。

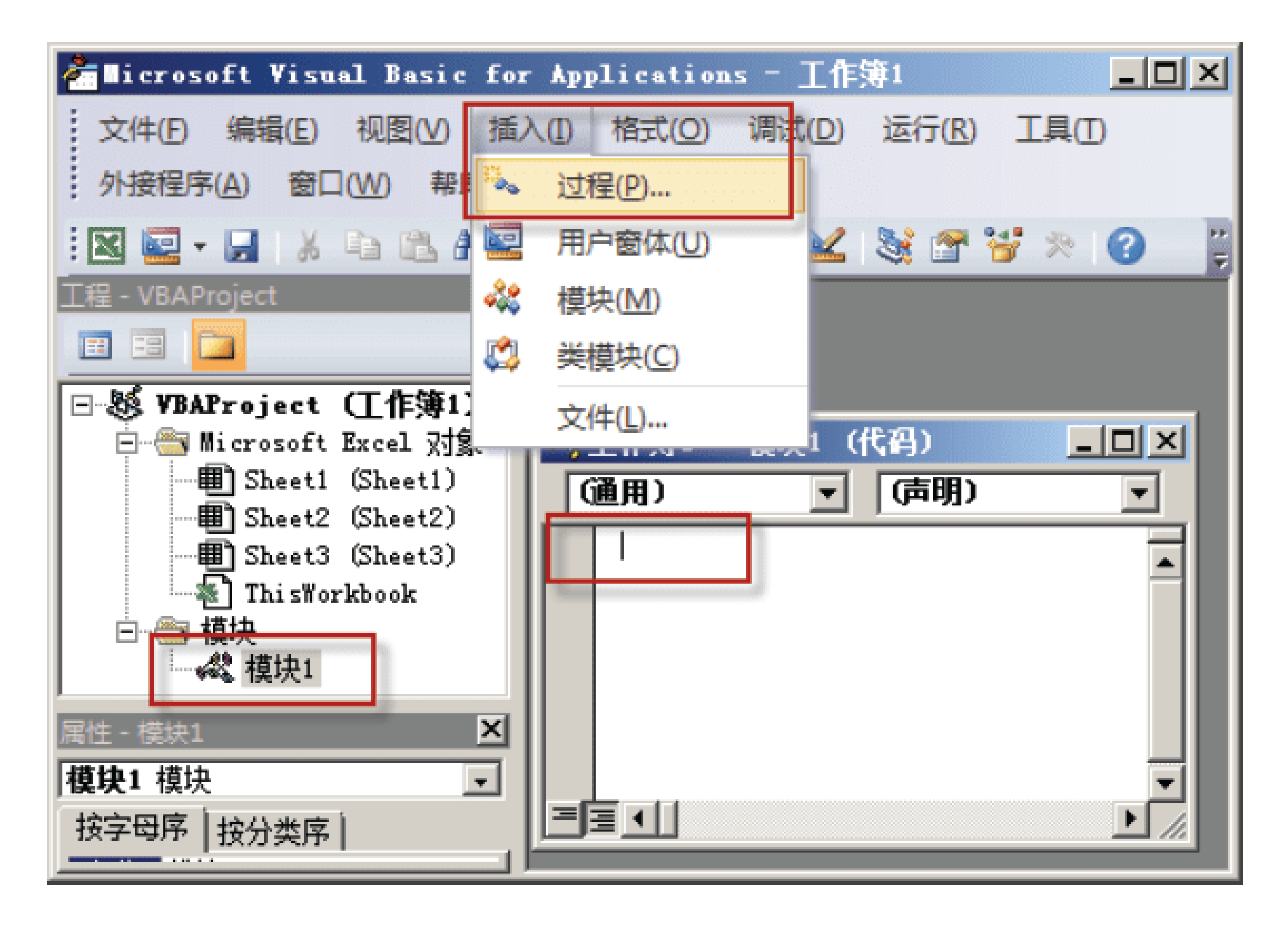


图 8.3　选择“插入”→“过程”命令

（3）单击“确定”按钮关闭“添加过程”对话框，Visual Basic 编辑器在模块的“代码”窗口中按照设置自动生成模块的结构代码，如图 8.5 所示。

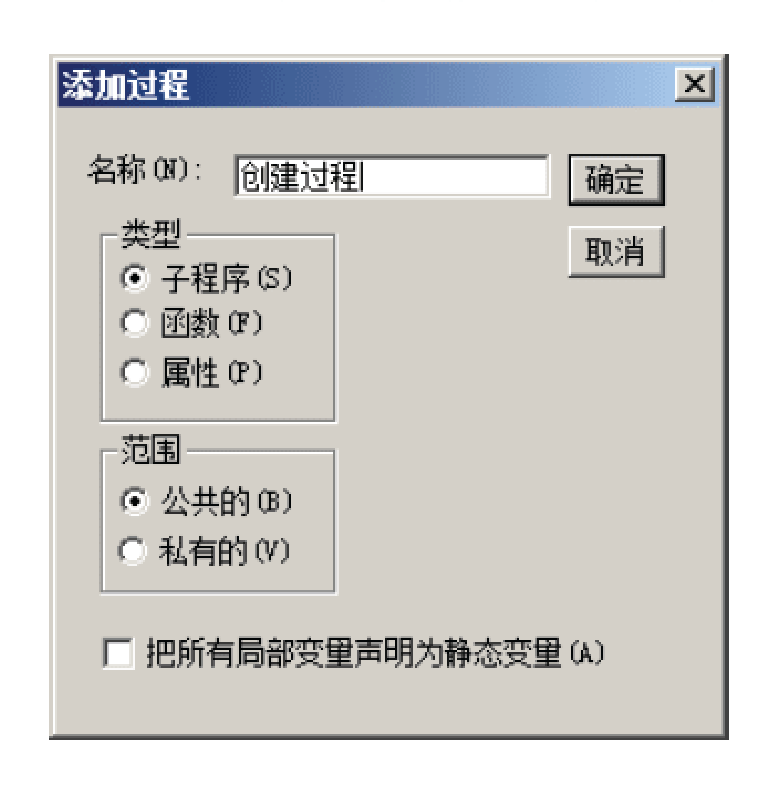


图 8.4　“添加过程”对话框

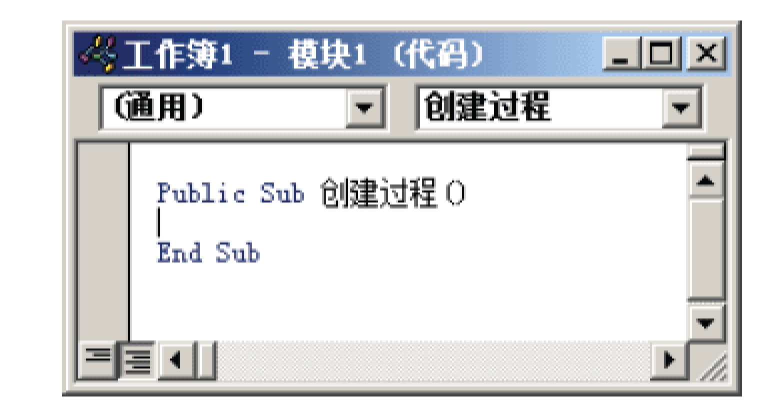


图 8.5　自动生成过程的结构代码

提示：对于熟练的程序设计人员来说，过程的创建没有这么复杂，可以直接在“代码”窗口中输入过程声明语句，如这里的“Public Static Sub 创建过程()”，按 Enter 键后 Visual Basic 编辑器会自动在添加“End Sub”语句创建一个过程结构。

8.2.2　调用 Sub 过程

在程序中使用过程能够将程序分成多个小的程序块，每块程序完成一个任务，最后通过组合这些过程完成需要的复杂任务。每一个过程在程序中起着一个服务的作用，当程序需要这个服务时，就要使用这个服务过程，这在 VBA 中称为调用过程。调用过程可以使用 Call 语句来实现，其语法格式如下：

```
Call 过程名（过程参数列表）
```

参数说明如下所示。

- ❑ 过程名：需要调用过程的名称。
- ❑ 过程参数列表：被调用过程中的参数。被调用过程如果有参数，该参数列表在调用过程时必须用括号括起来，如果过程没有参数，括号可以省略。

在调用过程时，关键字 Call 可以省略，直接使用过程名可实现对过程的调用，其语法格式如下所示。

```
过程名（过程参数列表）
```

在调用过程时，VBA 将程序执行的控制权转移给被调用的过程，当过程执行中遇到 End Sub 或 Exit Sub 时，控制权随之转移给调用程序的下一行继续执行。

【范例 8-1】 调用子过程计算货品销售统计表中货品销售总价，当表中缺少单价或件数值时，程序给出提示。代码如下所示。

```
Sub main()
    Dim v As Integer, c As Integer         '声明变量
    v = Cells(Selection.Row, 2)            '获得单价
    c = Cells(Selection.Row, 3)            '获得件数
    heji v, c                              '调用子过程计算总价
End Sub
Sub heji(a, b)                             '声明子过程
    Dim p As Integer                       '声明总价变量
    If a = 0 Or b =0 Then                  '是否缺少数据
        MsgBox "数据不全，无法计算！"          '缺少数据则给出提示
        Exit Sub                           '退出子过程
    End If
    p = a * b                              '计算总价
    Cells(Selection.Row, 4) = p            '写入单元格
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，程序计算所在行的货品单价和件数的积，如图 8.6 所示。如果所在行缺失单价和件数值，程序给出提示，如图 8.7 所示。

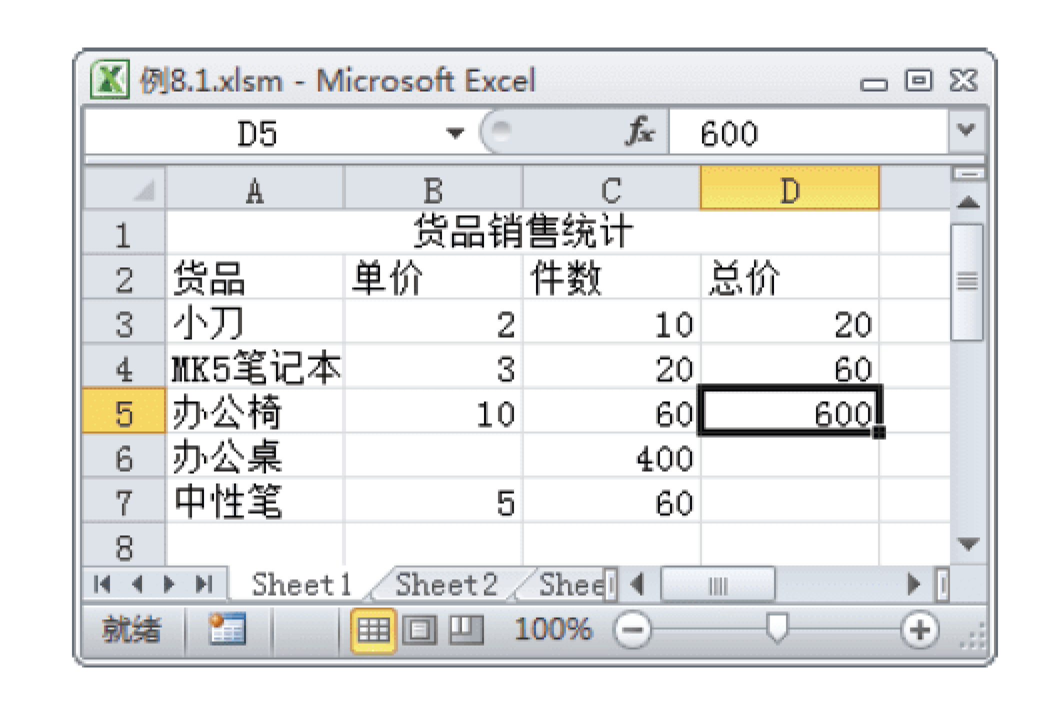


图 8.6　计算货品总价

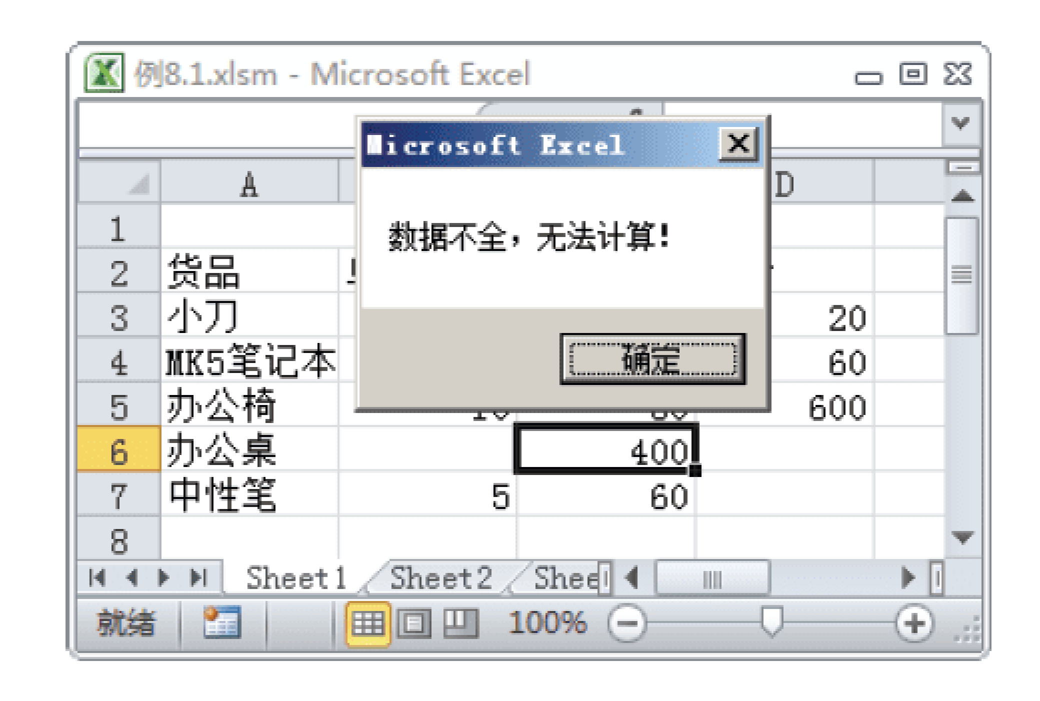


图 8.7　错误提示

【代码解析】本示例演示子过程调用的方法。在代码中，在 main 过程中直接调用 heji 子过程，过程名后带有省略括号的参数列表，即选择单元格所在行的第 2 列和第 3 列的单元格数据。heji 过程计算货品总价，并使用 If 语句判断参数单价和件数是否为 0，如果为 0，

则说明单元格中没有数据，给出提示；否则，则计算总价并写入总价单元格中。

8.3 VBA 中参数的传递

在调用过程时，过程间必然存在着数据的交流。就像 8.2.节的范例 8-1 中那样，要调用 heji 过程来求计算总价，必须告诉该过程计算总价的数据来自哪些单元格。在 VBA 中，过程间参数的传递分为按值传递和按地址传递这两种参数的传递方式。

8.3.1 使用地址参数传递

在介绍按地址传递参数前，首先需要了解形式参数和实际参数的概念。所谓形式参数，即在定义 Sub 过程时出现的变量名，因为没有具体的值，只是一个形式上的参数，所以称为形式参数。而实际参数是在 Sub 过程调用时进行传递的值。在 VBA 中实际参数可以是常量、变量、数组和对象类数据。

在 VBA 中，实际参数传递给形式参数的方式有两种，即按地址传递和按值传递。按地址传递参数是 VBA 默认的传递参数的方式。在定义过程时，如果形式参数前面加上了 ByRef 关键字，则该参数就将使用按地址的方式来进行传递。

按地址的传递方式，实际参数变量的地址被传递给形式参数，形式参数与实际参数都对应同一块内存区域。这样，在过程中形式参数的值改变，在返回调用程序后，实际参数也能够访问改变后的值。下面通过一个具体的范例来说明按地址的参数传递方式。

【范例 8-2】 对单元格修改，使程序能够在缺失数据的单元格中写入文字“N/A”进行提示，代码如下所示。

```
Sub main()
    Dim v As Variant, c As Variant                  '声明变量
    v = Cells(Selection.Row, 2)                     '获得单价
    c = Cells(Selection.Row, 3)                     '获得件数
    heji v, c                                       '调用子过程计算总价
    Cells(Selection.Row, 2) = v                     '单价单元格写入新数据
    Cells(Selection.Row, 3) = c                     '件数单元格写入新数据
End Sub
Sub heji(ByRef a As Variant,ByRef b As Variant) '声明子过程
    Dim p As Integer                                '声明总价变量
    If a = "" Then                                  '如果缺少单价
        MsgBox "数据不全，无法计算！"               '给出提示
        a = "N/A"                                   '变量赋值
        Exit Sub                                    '退出子过程
    End If
    If b = "" Then                                  '如果缺少件数
        MsgBox "数据不全，无法计算！"               '给出提示
        b = "N/A"                                   '变量赋值
        Exit Sub                                    '退出子过程
    End If
    p = a * b                                       '计算总价
    Cells(Selection.Row, 4) = p                     '写入单元格
```

```
23  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口中输入以上代码，按 F5 键运行程序，程序运行效果与范例 8-1 基本相同，只是当单价或件数缺失时，关闭提示对话框中，可以在缺失的单元格中写入“N/A”字样以示提示，如图 8.8 所示。

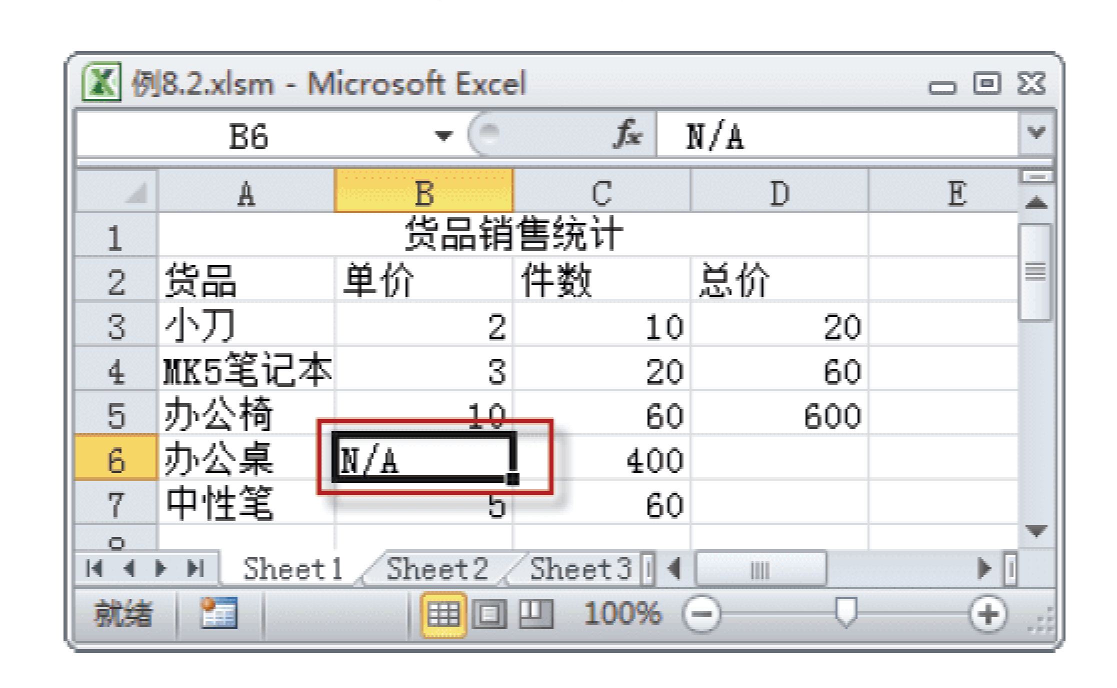


图 8.8　“立即窗口”显示各个参数的值

【代码解析】本段示例代码在进行过程调用时，使用按地址方式实现产生的传递。与范例 8-1 相比，在第 09 行声明子过程时设置好参数传递方式为按地址的传递方式。在子过程中代码的第 11～15 行和第 16～20 行分别对单价和件数单元格的内容进行判断，如果为空，参数赋值为“N/A”。在主程序的第 06 行和第 07 行，在完成子过程调用后单元格中写入变量 v 和 c 的值，由于是按地址传递参数，这两个变量的值与子过程中变量值一致。另外，由于变量的值即可能为数字，又可能是字符，变量都声明为 Variant 型。

注意：当使用按地址的参数传递方式时，只有当实际参数是变量时，才能够进行参数的传递。如果实际参数是一个常量或表达式，参数将无法采用按地址的方式传递。

8.3.2　使用值参数传递

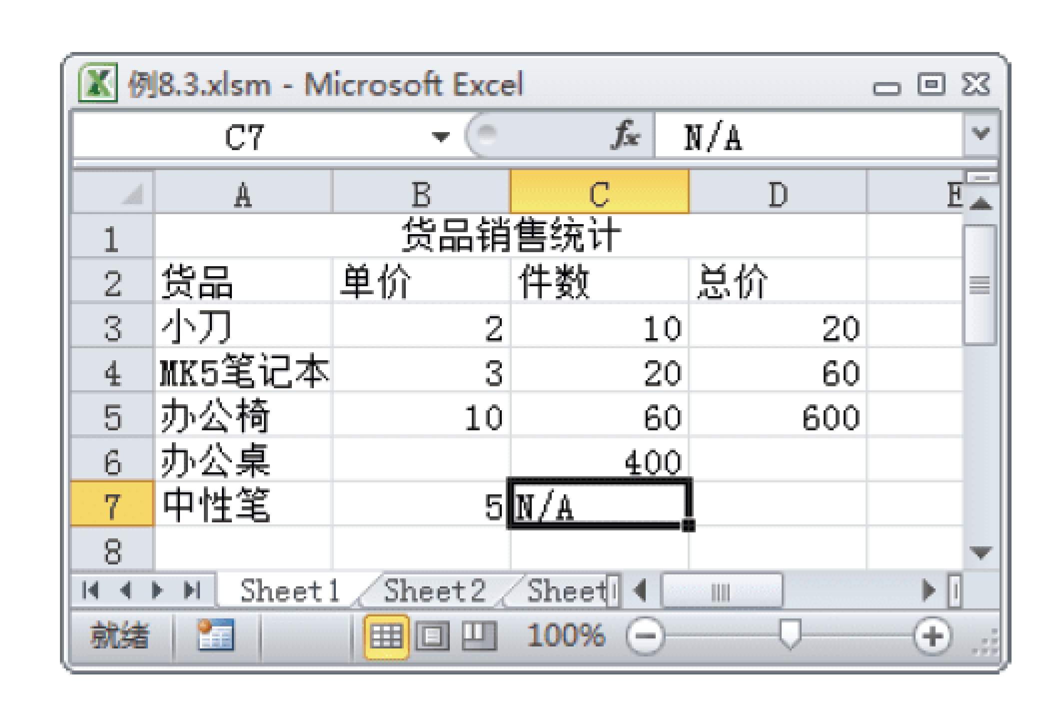
按值传递参数时，实际参数的值作为副本将直接赋值给形式参数。在过程的定义过程中，在形式参数前添加 ByVal 关键字，则参数将按传值的方式传递。下面将范例 8-2 改为按值传递参数的方式来比较其与按地址传递参数的方式的不同。

【范例 8-3】 修改范例 8-2 的代码，使用按值传递参数标记空白单元格，代码如下所示。

```
Sub main()
    Dim v As Variant, c As Variant                    '声明变量
    v = Cells(Selection.Row, 2)                       '获得单价
    c = Cells(Selection.Row, 3)                       '获得件数
    heji v, c                                         '调用子过程计算总价
End Sub
Sub heji(ByVal a As Variant, ByVal b As Variant)     '声明子过程
    If a = "" Then                                    '如果缺少单价
        MsgBox "数据不全，无法计算！"                  '给出提示
        Cells(Selection.Row, 2) = "N/A"               '单价单元格写入标示
        Exit Sub                                      '退出子过程
    End If
    If b = "" Then                                    '如果缺少件数
        MsgBox "数据不全，无法计算！"                  '给出提示
```

```
15            Cells(Selection.Row, 3) = "N/A "          '件数单元格写入新标示
16            Exit Sub                                   '退出子过程
17        End If
18        p = a * b                                      '计算总价
19        Cells(Selection.Row, 4) = p                    '写入单元格
20    End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，程序运行效果与范例 8-2 相同，当单价或件数单元格为空时，给出提示后标示为空的单元格，如图 8.9 所示。



	A	B	C	D
1	货品销售统计			
2	货品	单价	件数	总价
3	小刀	2	10	20
4	MK5笔记本	3	20	60
5	办公椅	10	60	600
6	办公桌		400	
7	中性笔	5	N/A	
8				

图 8.9　程序运行效果

【代码解析】本段示例程序在调用过程时，采用传值方式来传递参数。在第 07 行将声明子过程时，将参数方式定义为按值传递。此时，完成子过程调用后，主程序中的变量 v 和 c 的值不会更改为子过程中对应变量 a 和 b 的值，使用完全相同的代码将无法更新单价和件数单元格中的内容。此时的解决方法是直接在子过程中向单元格写入标示文字。

提示：在过程中传递变量时，按地址的传递方式比按值的传递方式效率更高，但是传地址方式中的形式参数不是一个真正的局部变量，其值在调用子过程后会改变。这也许会对程序产生影响，因此应尽量使用按值的传递方式。

8.3.3　使用数组参数传递

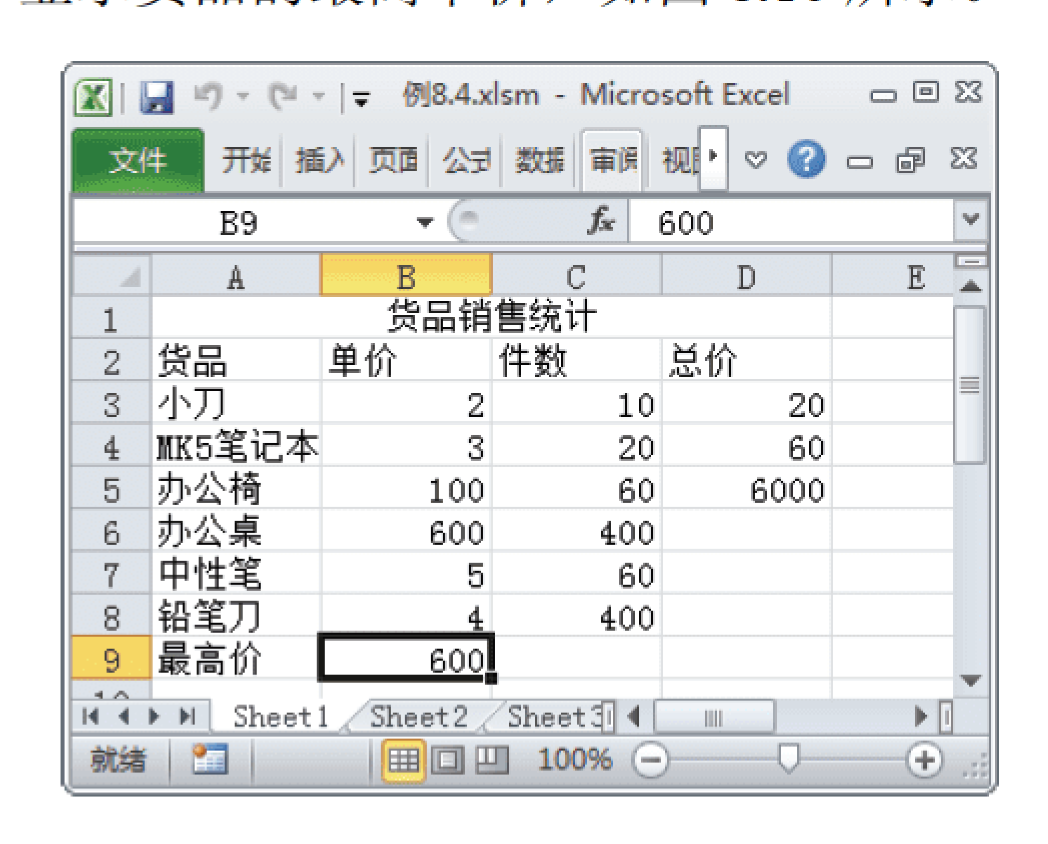
数组是一组变量的集合，其在内存中是一系列的连续的区域。在进行过程调用时，数组同样可以作为参数传递给子过程进行处理。数组的传递一般都是使用按地址的传递方式来实现，具体的操作见下面的范例 8-4。

【范例 8-4】 编程计算范例 8-3 的货品销售统计表中单价的最大值，代码如下所示。

```
Sub main()
    Dim n As Integer, myArray(5) As Integer     '声明数组和变量
    For n = 3 To 8                              '循环计数
        myArray(n - 3) = Cells(n, 2)            '将单价赋予数组
    Next
    myMax myArray()                             '调用子过程
End Sub
Sub myMax(a() As Integer)                       '声明子过程
    Dim i As Integer, m As Integer              '声明变量
```

```
10        m = a(LBound(a))                         '数组中第一个数值赋予变量
11        For i = LBound(a) To UBound(a)           '遍历数组中的所有数值
12            If a(i) > m Then m = a(i)            '大数赋予变量
13        Next
14        Cells(9, 2) = m                          '结果写入指定单元格
15    End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，在工作表的 B9 单元格中显示货品的最高单价，如图 8.10 所示。



	A	B	C	D
1	货品销售统计			
2	货品	单价	件数	总价
3	小刀	2	10	20
4	MK5笔记本	3	20	60
5	办公椅	100	60	6000
6	办公桌	600	400	
7	中性笔	5	60	
8	铅笔刀	4	400	
9	最高价	600		

图 8.10　程序运行后显示最高单价

【代码解析】本段程序演示调用子过程时数组参数的传递方法。在 main 过程中，第 02～05 行代码使用 For...Next 循环来遍历单价栏中的所有货品单价值，将其赋予数组。子过程中比较数组元素值的大小，将最大值放置于变量 m 中。在找到最大值后将这个值写入指定单元格中。

注意：*在子过程中声明数组变量时要注意，在定义数组形式参数时，输入的数组名后必须跟上空的括号。*

8.3.4　使用可选参数

对于大多数应用程序来说，在调用子过程，形式参数和实际参数间的数据传递都是按照位置来实现的。也就是说，调用 Sub 过程所使用的实际参数的次序与定义 Sub 过程中定义的形式参数的次序是相对应的。但有时候，会出现无法确定实际参数是否真正传递的情况，这时可以在定义形式参数时，将形式参数定义为可选参数。

在 VBA 的过程中，可以通过在形式参数前面添加关键字 Optional 来将形式参数定义为可选参数。同时，在子过程内部，可以通过 IsMissing 函数来测试调用程序是否真的有参数传递给了这个可选参数。关于可选参数的用法，下面使用一个范例来说明。

【范例 8-5】 使用输入对话框向职工基本信息表中输入职工姓名、性别、年龄和身份证号信息。这里，身份证号为可选输入，如果不输入，对应单元格空白。其他信息为必输入内容，如果未输入，退出过程。本范例的详细代码如下所示。

```
Sub main()
    Dim p As String, q As String, r As String          '声明变量
```

```
03      p = InputBox("请输入姓名")                        '输入姓名
04      If p="" then Exit Sub                            '未输入姓名退出过程
05      q = InputBox("请输入性别")                        '输入性别
06      If q="" then Exit Sub                            '未输入性别退出过程
07      r = InputBox("请输入年龄")                        '输入年龄
08      If r="" then Exit Sub                            '未输入性别退出过程
09      s = InputBox("请输入身份证号")                    '输入身份证号
10      If s = "" Then                                   '如果没有输入身份证号
11          luru p, q, r                                 '调用子过程传递 3 个参数
12      Else
13          luru p, q, r, s                              '否则调用子过程传递 4 个参数
14      End If
15  End Sub
16  Sub luru(n As String, s As String, a As String, Optional i)'声明子过程
17      With ActiveSheet
18          .Range("a3") = n                             '向单元格中写入姓名
19          .Range("b3") = s                             '向单元格中写入性别
20          .Range("c3") = a                             '向单元格中写入年龄
21          If IsMissing(i) Then                         '是否有身份证号
22              .Range("d3") = ""                        '没有则该单元格为空
23          Else
24              .rang("d3") = i                          '有则写入单元格
25          End If
26      End With
27  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，程序给出输入对话框，依次输入姓名、性别、年龄和身份证号信息后，信息被写入工作表中，如图 8.11 所示。基本信息项目中如果有任意一项没有输入，则程序退出，前面的输入内容不会写入工作表。如果只是身份证号未输入，退出过程时前面输入内容会写入工作表中。

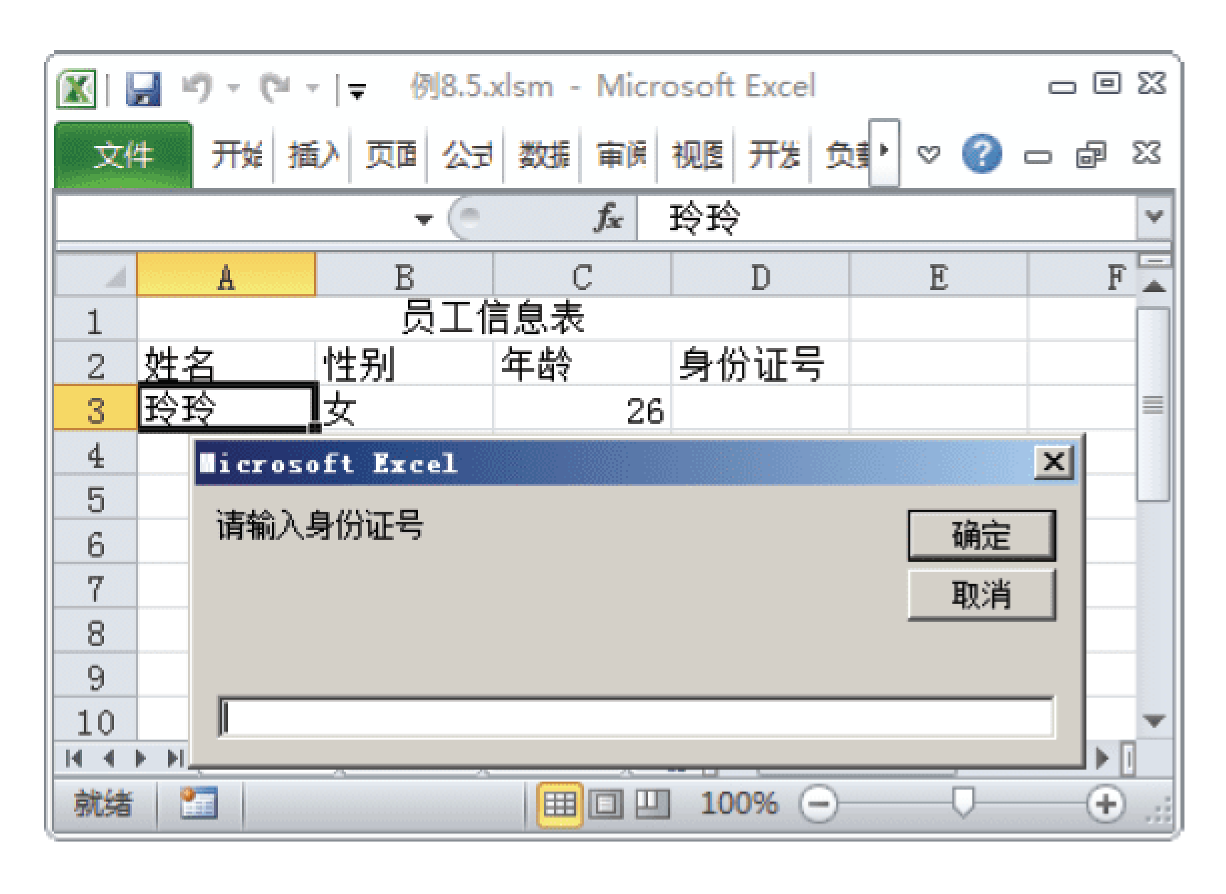


图 8.11　使用输入对话框输入员工基本信息

【代码解析】本示例演示子过程调用时可选参数的使用方法，这里的身份证号是可选参数。本示例 main 过程的程序流程图，如图 8.12 所示。程序中，使用 InputBox 函数输入用户基本信息。代码的第 03～09 行判断输入情况，输入框没有输入任何值，程序退出。第 09～14 行判断身份证号是否输入，如果输入则调用过程时传递该参数给子过程，否则则不传递该参数。在第 16 行声明子过程时使用 Option 来定义可选参数。第 21 行代码使用

IsMissing 函数来判断是否传递了该参数，根据不同的情况决定身份证号单元格的内容。在子过程中，ActiveSheet 表示当前活动工作表，使用 With 结构来为活动工作表的指定单元格赋值。

注意：在过程中可以定义多个可选参数，这些可选参数必须放在参数列表的最后面，它们的数据类型是 Variant 型。

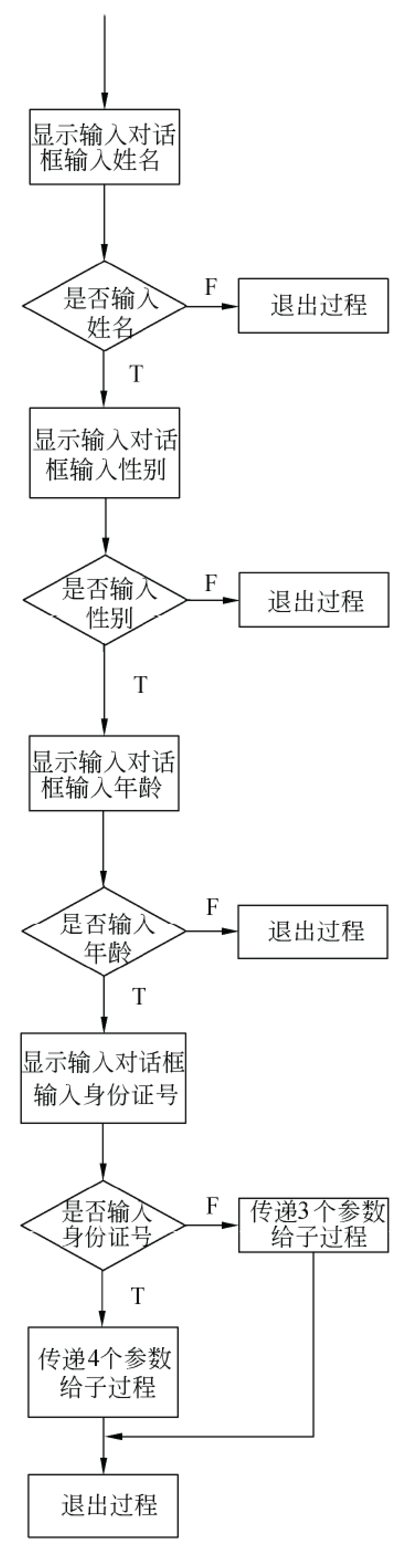


图 8.12　示例 main 过程的流程图

8.3.5　使用可变参数

有时候在调用子过程时，参数的个数在定义时还无法确定。此时，可以在定义过程的参数表最后一个参数前加上 ParamArray 关键字，这样子过程可以接受任意个数的参数。下面通过一个范例来看看可变参数的使用方法。

【范例 8-6】 对数据修改，使退出程序时前面输入的内容能够写入工作表，代码如下所示。

```
Sub main()
    Dim p As String, q As String, r As String       '声明变量
    p = InputBox("请输入姓名")                      '输入姓名
    If p = "" Then
        Exit Sub                                    '未输入退出过程
    End If
    q = InputBox("请输入性别")                      '输入性别
    If q = "" Then                                  '如果没有输入性别
        luru p                                      '调用过程写入姓名
        Exit Sub                                    '退出过程
    End If
    r = InputBox("请输入年龄")                      '输入年龄
    If r = "" Then                                  '如果没有输入年龄
        luru p, q                                   '调用过程写入姓名和性别
        Exit Sub                                    '退出过程
    End If
    s = InputBox("请输入身份证号")                  '输入身份证号
        If s = "" Then                              '没有输入身份证号
        luru p, q, r                                '调用子过程写入姓名、性别和年龄
        Exit Sub                                    '退出过程
    End If
    luru p, q, r, s                                 '调用过程写入所有信息
End Sub
Sub luru(n As String, ParamArray others()) '声明子过程
    Dim i As Integer                                '声明变量
    ActiveSheet.Range("a3") = n                     '向单元格中写入姓名
    For i = LBound(others) To UBound(others)    '变量可变参数
        Cells(3, i + 2) = others(i)             '写入单元格
    Next
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序。与范例 8-5 不同的是，在输入框中未输入任何内容而关闭输入框，能够将前面输入的内容写入工作表中，如图 8.13 所示。

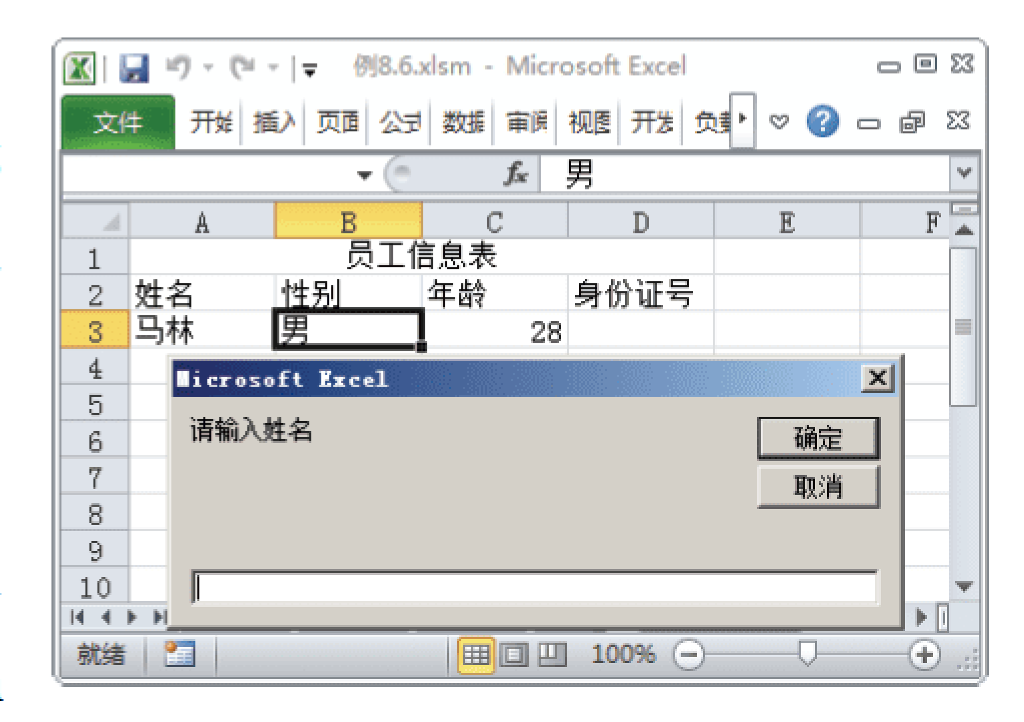


图 8.13　程序运行效果

【代码解析】本示例演示可变参数的使用方法。要达到设计要求，由于无法确定用户输入数据的个数，只能使用可变参数来传递数据。本示例的 main 过程流程图如图 8.14 所示。在 main 过程中，使用

If 语句来判断输入框中的输入情况，根据不同的输入来确定调用子过程时应该传递的参数。在第 24 行子过程声明时，使用 ParamArray 关键字定义了一个可变参数。这个可变参数是一个数组，在代码的第 27 行以该数组的上标和下标作为循环条件，通过循环来遍历所有参数，将参数值写入对应单元格中。

注意： ParamArray 关键字的使用和 Opitional 关键字的使用一样，也是只能用于参数列表中最后一个参数，这里定义的实际上是一个 Variant 类型的 Optional 数组。另外，ParamArray 关键字不能和 ByVal、ByRef 或 Optional 关键字一起使用。

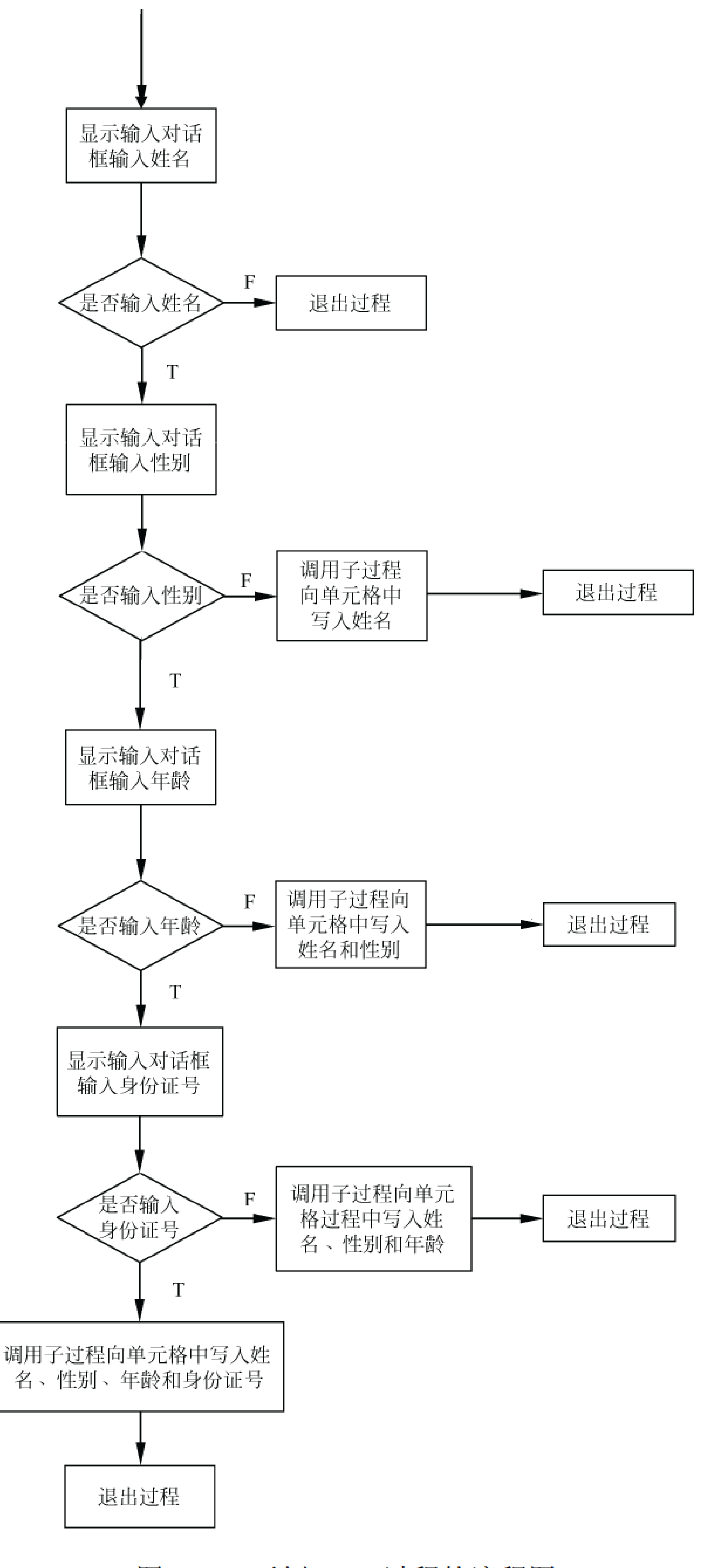


图 8.14　示例 main 过程的流程图

8.4　使用 Function 过程

VBA 本身提供了很多的内置函数，同时 VBA 也可以调用各种工作表函数，这些都使得 VBA 在进行各类数据处理的过程中具有很强的能力。但即便如此，这些函数也不能满足所有的编程需要，此时可以通过编写 Function 过程来创建自己的函数，使其能够像 VBA 内部函数一样的使用。在 VBA 中，Function 过程和 Sub 过程一样，是通用过程的一种，Function 过程也称为 Function 函数。

8.4.1　理解函数

在程序设计语言中，函数是为了完成特定功能而创建的一个过程。在程序中对函数的使用称为调用函数。在 Excel VBA 中，可以使用的函数包括 VBA 自带的函数和 Excel 工作表函数。在 VBA 程序中调用函数和调用过程的方法相同，其一般的语法格式如下：

```
函数名 （参数列表）
```

这里，参数列表中的参数也称为自变量，参数放在圆括号中，多个参数使用逗号来进行分隔。在 VBA 中，还可以使用 WorkSheetFunction 对象来使用 Excel 工作表函数，其语法格式如下：

```
Application.WorkSheetFunction 函数名 （参数列表）
```

例如，需要求单元格区域 A1：A10 数据的最大值，可以使用工作表函数 Max 来实现，调用该函数的程序代码如下所示。

```
x=Application.WorkSheetFunction Max("A1:A10")
```

在 Excel 中使用 VBA 编写程序时，往往需要将一个函数的返回值赋予表格中的单元格。此时，可以指定函数作为 Range 对象的 Formula 属性。如，计算 A2 至 A10 单元格中数据的和，并将结果赋予 A11 单元格，可采用下面的语句来实现：

```
WorkSheets（"Sheet1"）.Range（"A11"）.Formula="sum("A1:A9")
```

Excel VBA 的内置函数能分为 4 类，即算术函数、字符串函数、转换函数和日期/时间函数。下面使用一个使用字符串函数的范例来介绍 VBA 内置函数的使用。

【范例 8-7】 使用 VBA 函数来将选择单元格中字符的首字母自动改为大写，代码如下所示。

```
Sub 函数的使用()
    Dim a As String, b As String                    '声明变量
    If Selection.Count > 1 Then                     '如果选择多个单元格
        MsgBox "选择单元格个数超过 1，程序将退出", _
         vbOKOnly, "提示"                           '给出提示
        Exit Sub                                    '退出过程
    End If
    If IsEmpty(Selection) Then                      '如果选择单元格为空
```

```
            MsgBox "单元格为空，程序将退出",
            vbOKOnly, "提示"'                      '给出提示
            Exit Sub                               '退出过程
        End If
        a = Selection                              '获得当前选择单元格内容
        b = Left(a, 1)                             '获得选择单元格的第一个字符
        If Asc(b) >= 65 And Asc(b) <= 90 Then      '字符是否是大写
            MsgBox "首字母已经是大写", vbOKOnly, "提示"    '大写则给出提示
        ElseIf Asc(b) >= 97 And Asc(b) <= 122 Then     '如果是字母，进行转换
            Selection = UCase(b) & Mid(a, 2, Len(a))   '将第一个字母变为大写
        Else
            MsgBox "第一个字符不是字母", vbOKOnly, "提示" '字符不是字母，给出提示
        End If
    End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，如果在工作表中选择了多个单元格，程序给出提示，如图 8.15 所示。如果单元格为空，程序给出提示，如图 8.16 所示。

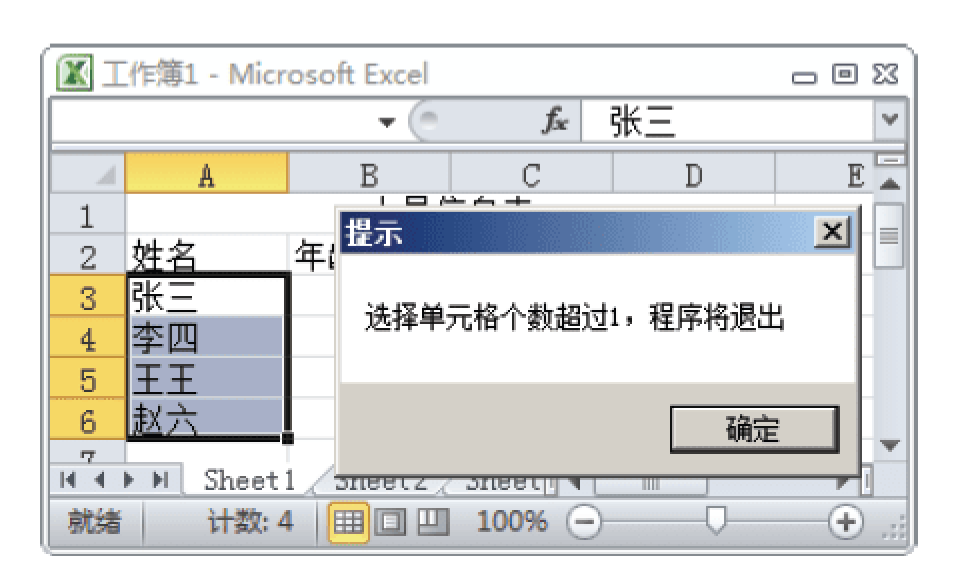


图 8.15　提示选择多个单元格

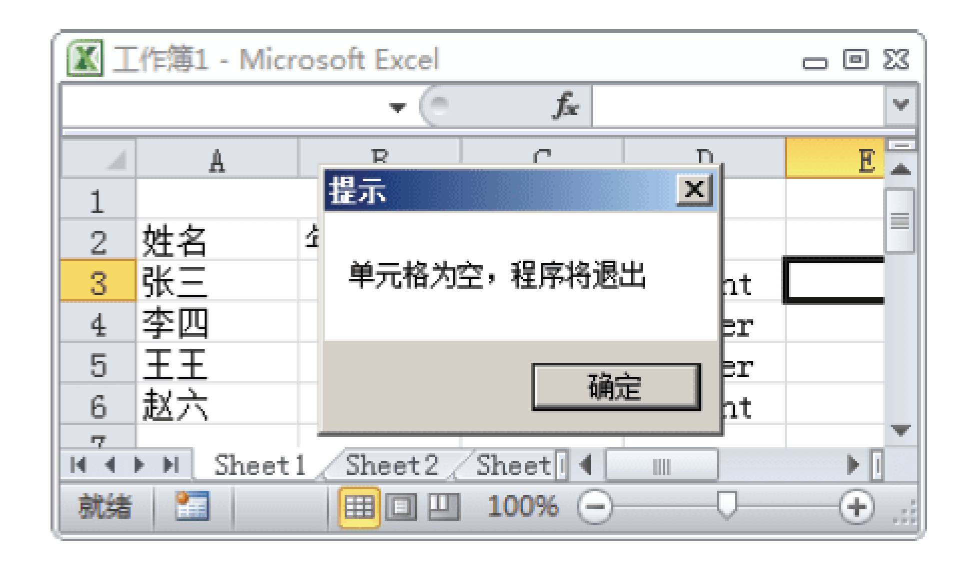


图 8.16　提示单元格为空

如果单元格中第 1 个字符不是字母，程序给出提示，如图 8.17 所示。当单元格中第 1 个字母已经是大写时，程序给出相应的提示，如图 8.18 所示。

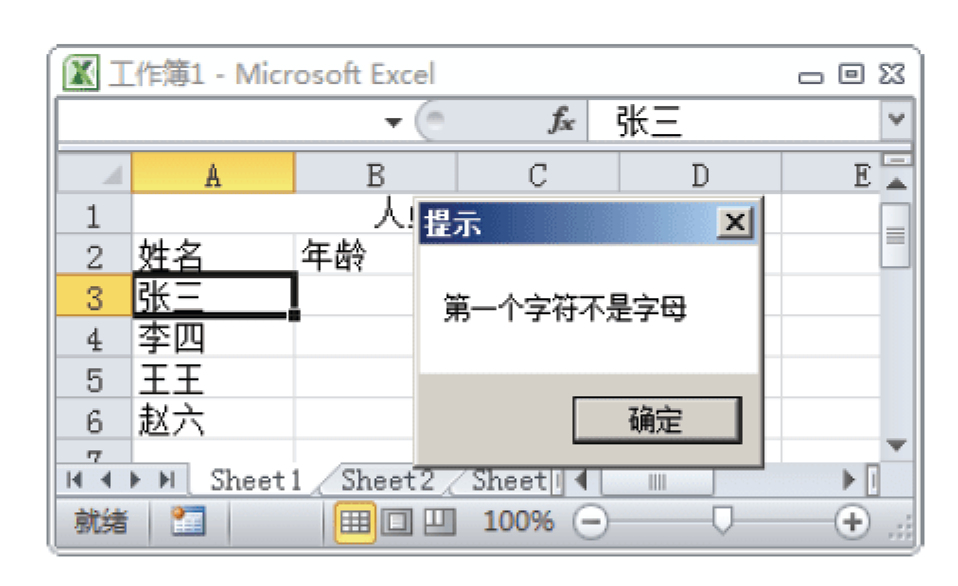


图 8.17　提示第一个字符不是字母

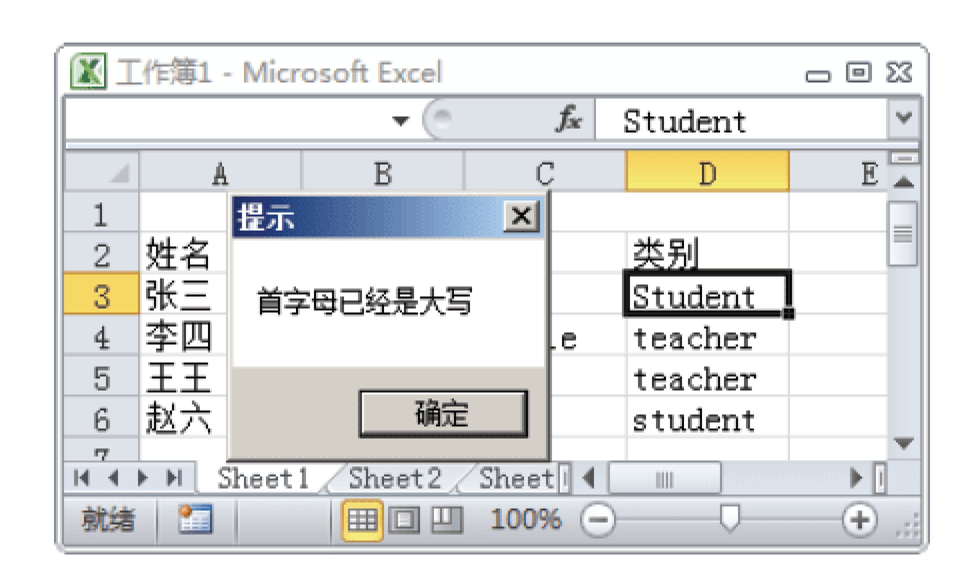


图 8.18　提示首字母已经大写

当单元格中第一个字符是小写字母时，程序自动将其转换为大写字母，如图 8.19 所示。

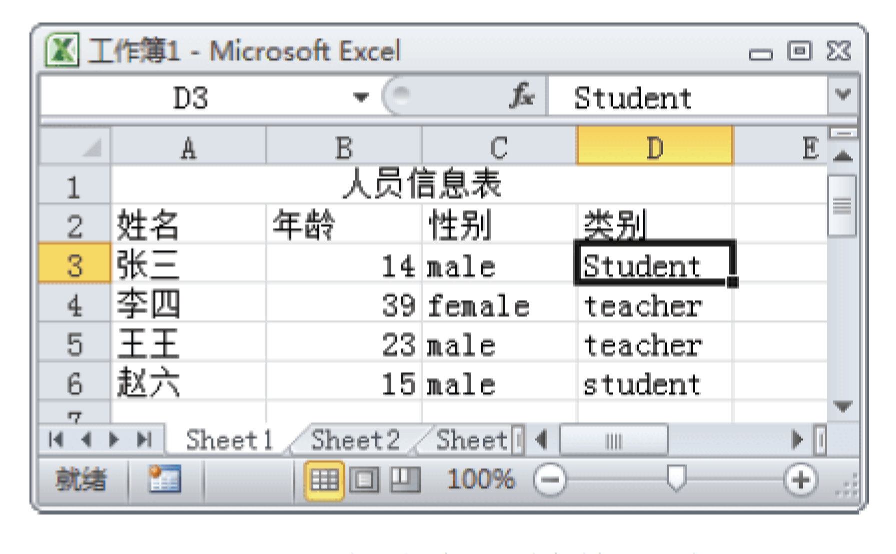


图 8.19　将首字母转换为大写

【代码解析】本段代码用于将单元格的第一个字母变为大写。程序运行时，首先判断选择是否选择了多个单元格，如果是则给出提示并退出过程。如果选择的是单个单元格，则判断该单元格是否为空，如果为空则给出提示并退出过程。在完成上述判断后，将选择单元格内容赋予变量 a，使用 Left 函数获得字符左侧第一个字母，判断该字母是否为大写，若是大写则给出提示，如是小写字母则进行转换，否则提示非字母并退出过程。本程序的流程如图 8.20 所示。

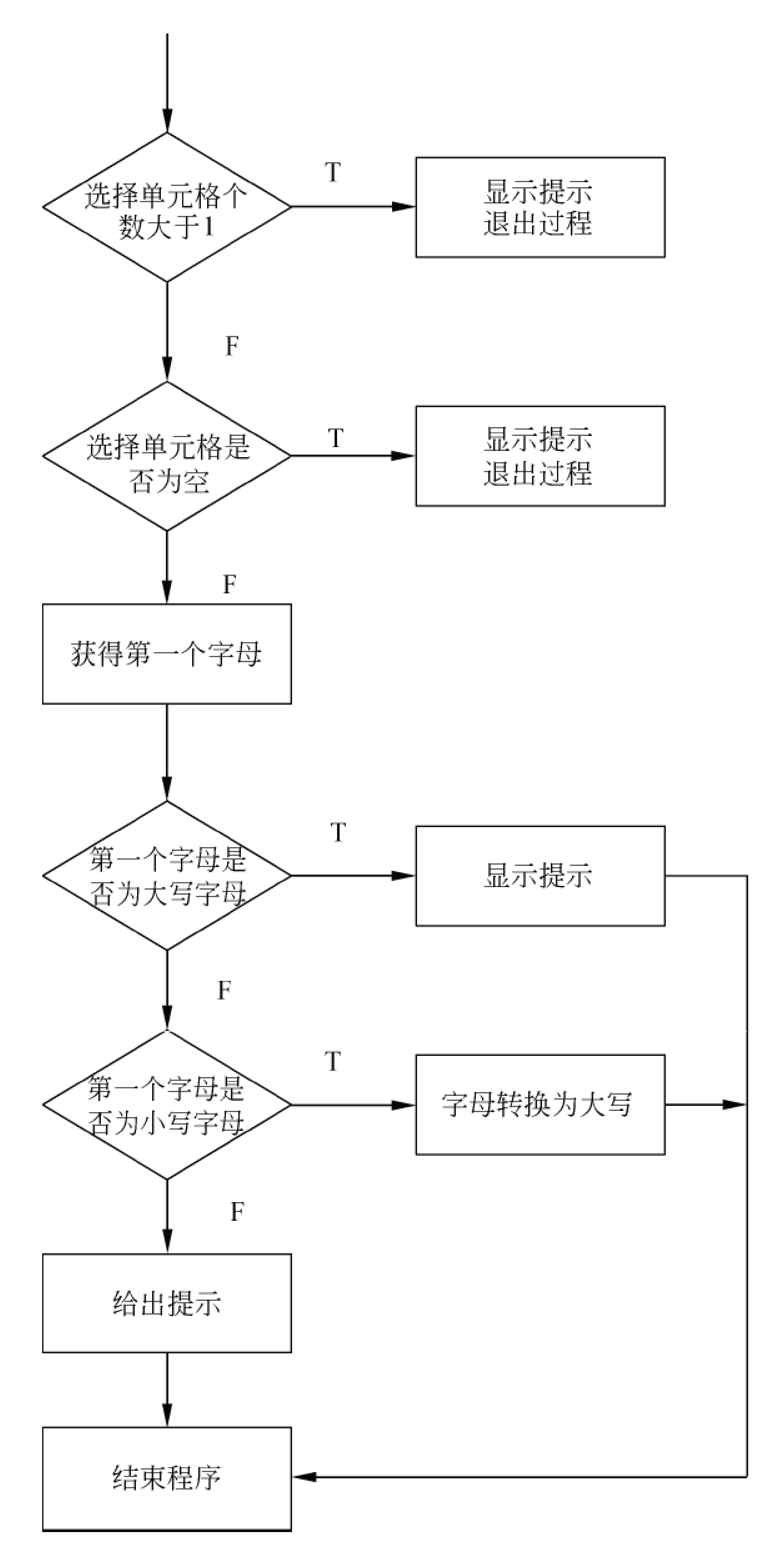


图 8.20　示例流程图

在程序的第 14 行使用了 Left 函数，该函数是 VBA 字符串函数，用于取字符串左侧的第 n 个字符。与其相对应的 Right 函数用于取字符串右边的第 n 个字符。Left 函数的语法格式为：

```
Left(字符串，n)
```

在程序的第 15 行使用了 Asc 函数，该函数为 VBA 转换函数，用于将字符转换为 ASCII 码值。大写字母的 ASCII 值在 65～90 之间，小写字母的 ASCII 值在 97～122 之间。如，Asc（"A"）的值为 65。

在程序的第 18 行使用了 Mid 函数，该函数是 VBA 字符串函数，用于获取从字符串的位置 p 到位置 n 间的所有字符。这里，如果省略了函数 q，则将从 p 开始选取后面的所有字符。Mid 函数语法格式为：

```
Mid(字符串,p,n)
```

在程序的第 18 行同时用到了 Ucase 函数，该函数是 VBA 转换函数，用于将小写字母转换为大写字母。与之相对应的 Lcase 函数将大写字母转换为小写字母。Ucase 函数的语法格式为：

```
Ucase(字符串)
```

提示：VBA 的内置函数十分丰富，限于篇幅，本书不进行详细介绍，读者可以使用 VBA 帮助文档或通过网络来查询各个函数的具体使用详情。

8.4.2　定义 Function 过程

Function 过程的创建与 Sub 过程的创建相似，但与 Sub 过程不同的是，Function 过程可以直接返回一个值给调用程序使用。Function 过程可以直接通过在“代码”窗口中输入代码来创建，其语法格式如下：

```
[Public|Private|Friend|Stactic] Function 函数名 ([参数列表]) [As 数据类型]
    语句块
    函数名=表达式
    [Exit Function]
    语句块
End Function
```

从上面介绍的语法结构可以看到，Function 过程与 Sub 过程的定义方式基本相同，其不同主要体现在两个方面。首先，在 Function 函数声明时最后使用了 As 数据类型语句，该语句用于定义函数的返回值类型，而 Sub 过程中没有这样的语句。

在 Function 函数体内部需要通过“函数名＝表达式”这种方式给函数名赋值，赋值是为了返回计算结果。这种用于返回计算结果的语句，对于 Sub 过程来说是不需要的。

提示：如果函数体内没有这个语句，则函数返回默认值。当函数为数值函数时，默认值为 0。如果函数为字符串函数，默认值为空字符串。如果是 Variant 函数，默认值为 Empty。如果是返回对象引用的函数，默认值为 Nothing。

8.4.3　直接调用 Function 过程

在程序中调用 Function 过程有两种方法，一种是直接调用，一种是使用 Call 语句来调

用。直接调用就像使用 VBA 内部函数一样，直接使用过程名，后面跟上参数即可。使用 Call 语句调用与 Sub 过程的调用方法完全相同。

【范例 8-8】 通过函数提取单元格中夹在文字间的数字，根据这些数字计算货品总价，代码如下所示。

```
01  Sub main()
02      Dim a As Integer, b As Integer                '声明变量
03      a = getNumber(Cells(Selection.Row, 1))        '调用函数获得货品件数
04      b = getNumber(Cells(Selection.Row, 2))        '调用函数获得单价值
05      Cells(Selection.Row, 3) = a * b               '相乘得到总价并写入单元格
06  End Sub
07  Function getNumber(g As String)                   '声明函数
08      Dim i As Integer                              '声明变量
09      For i = 1 To Len(g)                           '遍历文字中所有字符
10          If IsNumeric(Mid(g, i, 1)) Then           '判断当前字符是否为数字
11              getNumber = getNumber & Mid(g, i, 1)'是数字则将数字连接起来
12          End If
13      Next
14  End Function
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，计算选择单元格所在行的货品总价，如图 8.21 所示。

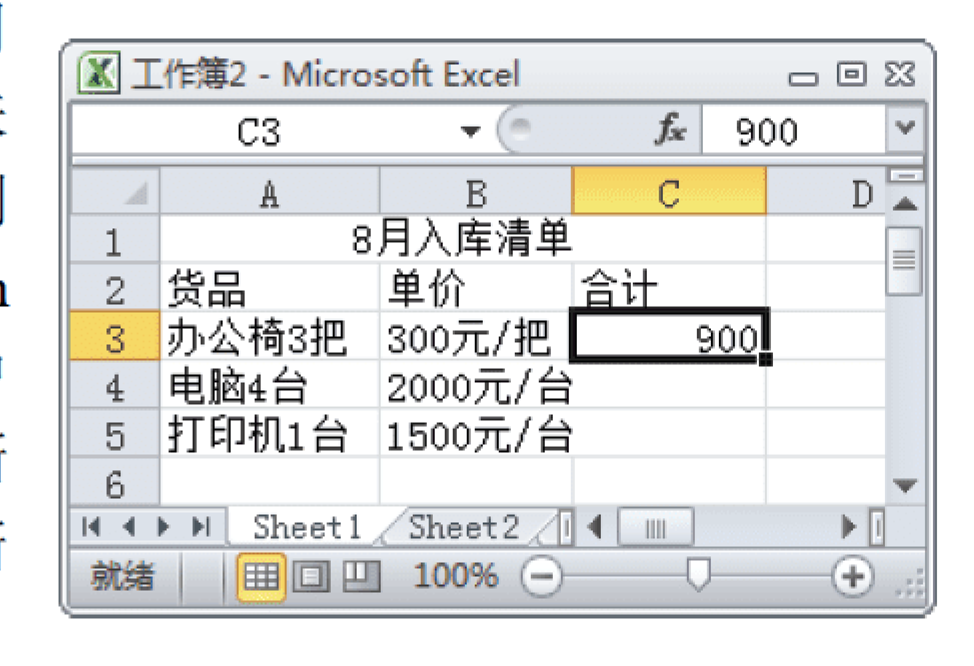


图 8.21　程序运行结果

【代码解析】本示例演示在过程代码中调用 Function 过程的方法。主程序调用 getNumber 函数来获取货品和单价中的数字，并将获得的数字相乘得到总价。程序的第 07 行声明 Function 过程，在 Function 过程中使用循环语句遍历单元格中所有字符，其中 Len（g）语句获得字符的总长度。第 10 行代码判断取出字符是否为数字，其中 IsNumeric 函数用于判断是否为数字，为数字时其值为 True。Mid（g，i，1）表示在字符串 g 中提取第 i 个字符，最后的参数 1 表示提取字符的个数为 1 个。程序的第 11 行使用连接运算符&将提取的数字连接为一个数字。

注意： 注意示例中反映出来的 Function 函数和 Sub 过程在使用上的不同。在程序的第 11 行通过给函数名赋值来获得计算结果。而在调用函数时，参数放在括号里面。

8.4.4　在 Excel 工作表中调用 Function 函数

自定义的 Function 函数和系统的内部工作表函数一样，可以在 Excel 工作表中以公式的形式进行引用。下面通过一个范例来介绍在工作表汇总调用函数的方法。本范例通过调用自定义 Function 函数计算一行数据在去掉最大值和最小值后的平均值，就像很多比赛中的评委评分那样。

（1）启动 Excel，打开包含数据的工作表。按“Alt+F11”键打开 VBA 编辑器。插入一个模块，在模块的“代码”窗口中输入如下程序代码：

```
01  Function avg(rng As Range)                                    '函数声明
02      Dim a As Single, b As Single, c As Single, d As Single, e As Single
                                                                  '声明变量
03      a = WorksheetFunction.Sum(rng)                            '计算平均值
04      b = WorksheetFunction.Min(rng)                            '获得最小值
05      c = WorksheetFunction.Max(rng)                            '获得最大值
06      d = WorksheetFunction.CountA(rng) -2                      '计算数据个数
07      e = (a - b - c) / d                                       '计算平均值
08      avg = e                                                   '返回计算结果
09  End Function
```

🔔**提示：** 这段代码创建了一个自定义函数。参数 rng 必须定义为 Range 对象类型，因为在后面调要用该函数，这个参数存储的是单元格对象。第 03～06 行，分别使用了 WorkSheetFunction 对象的 Sum 方法、Min 方法、Max 方法和 CountA 方法来获得选择单元格中数据的和、数据的最小值、数据的最大值和非空单元格的个数。在第 16 行之所以要减去 2，是因为本例中计算平均值去掉了最大和最小两个数值，数据的个数需要减去 2。

这里还要注意，在使用自定义函数时，应尽量引用工作表函数，工作表函数的计算速度大于自定义函数的速度，而且可以直接引用。

（2）返回 Excel 2010 窗口，选择单元格，单击“插入函数”按钮，如图 8.22 所示。

（3）在打开的“插入函数”对话框的“或选择类别”下拉列表框中选择“用户定义”选项，此时在“选择函数”列表中将列出所有的用户自定义函数。选择在步骤 1 中创建的函数，单击“确定”按钮，如图 8.23 所示。

（4）在打开的“函数参数”对话框中单击按钮，在数据表中拖动鼠标圈出参数所在的单元格，如图 8.24 所示。再次单击该按钮后将“函数参数”对话框重新展开，此时在对话框中已经可以看到选择的数据和计算结果了，如图 8.25 所示。

（5）单击“确定”按钮关闭“函数参数”对话框。单元格中显示使用自定义函数计算的结果。复制函数到其他的单元格中，对其他行的数据进行计算。本例的最终效果如图 8.26 所示。

🔔**提示：** 自定义函数的使用和 Excel 2010 内置函数的使用方法完全一样。除了上面调用自定义函数的方法外，还可以在单元格或公式的“编辑栏”中直接输入“=avg(B2:H2)”。完成输入后，按“Ctrl+Shift+Enter”键即可。

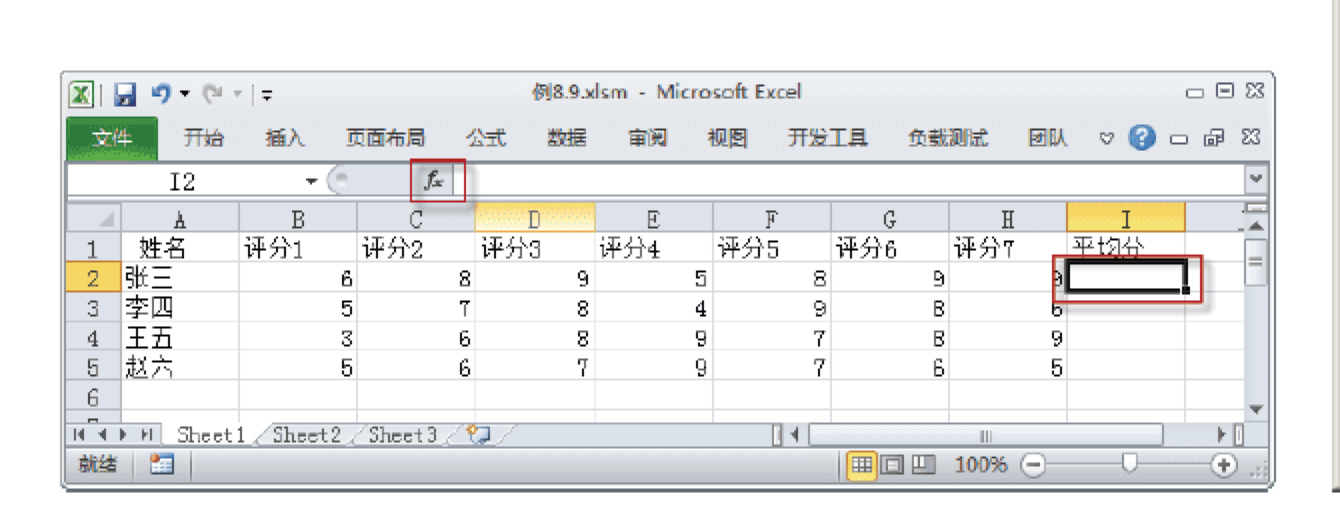


图 8.22　单击“插入函数”按钮

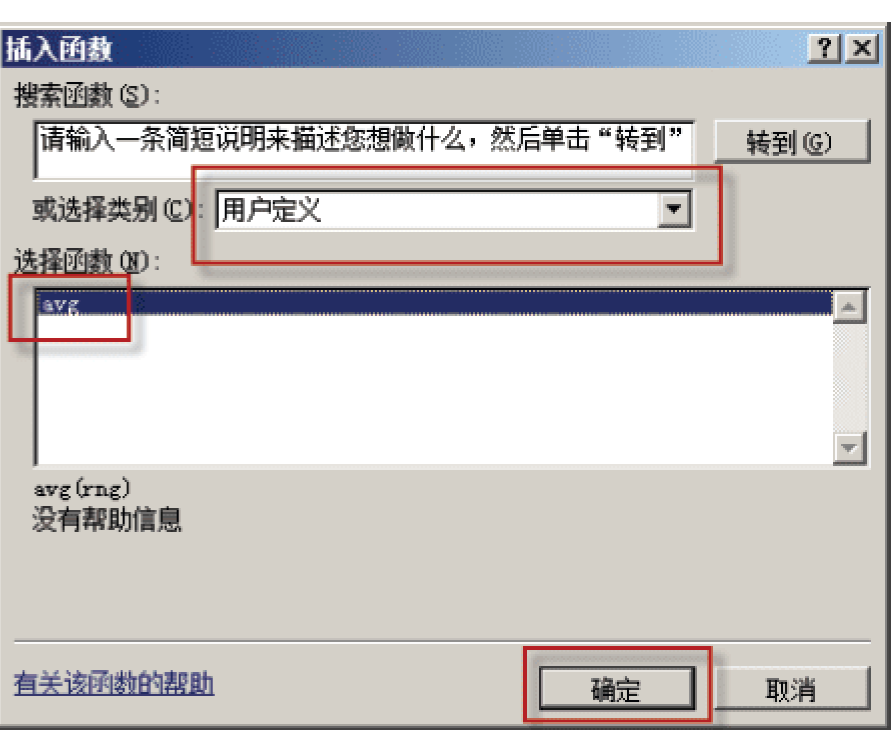


图 8.23　选择自定义函数

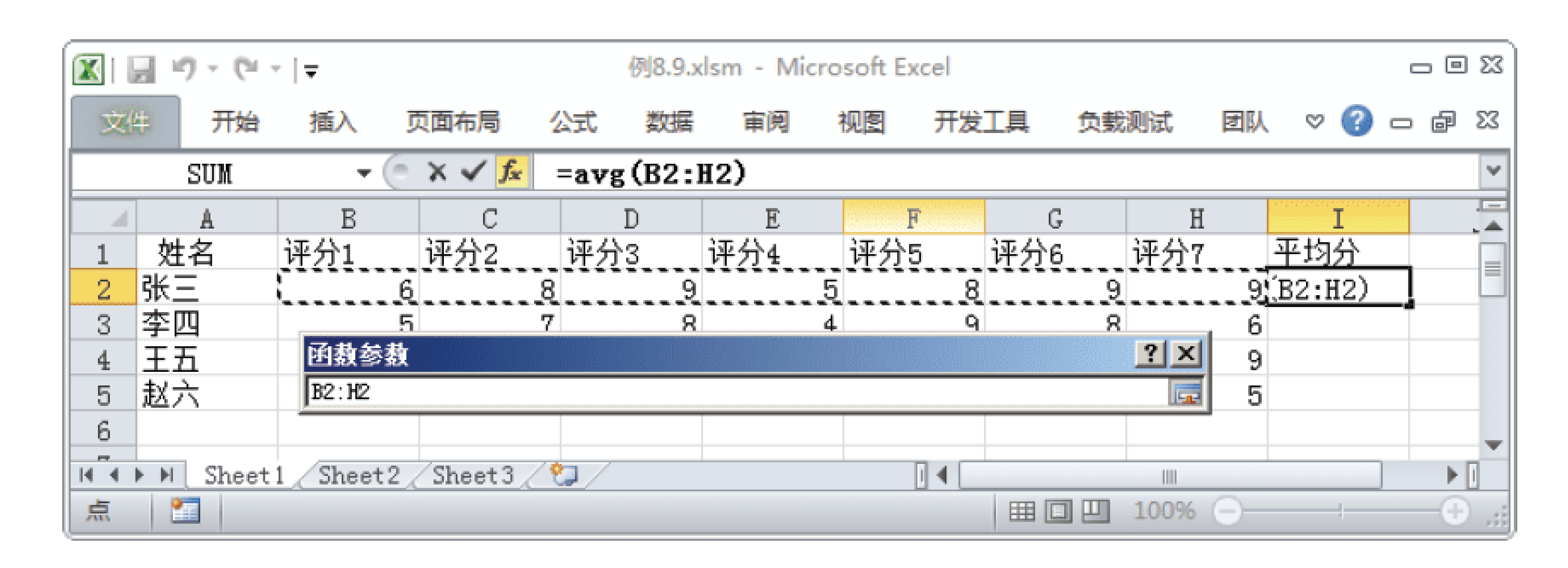


图 8.24　圈出参数所在的单元格

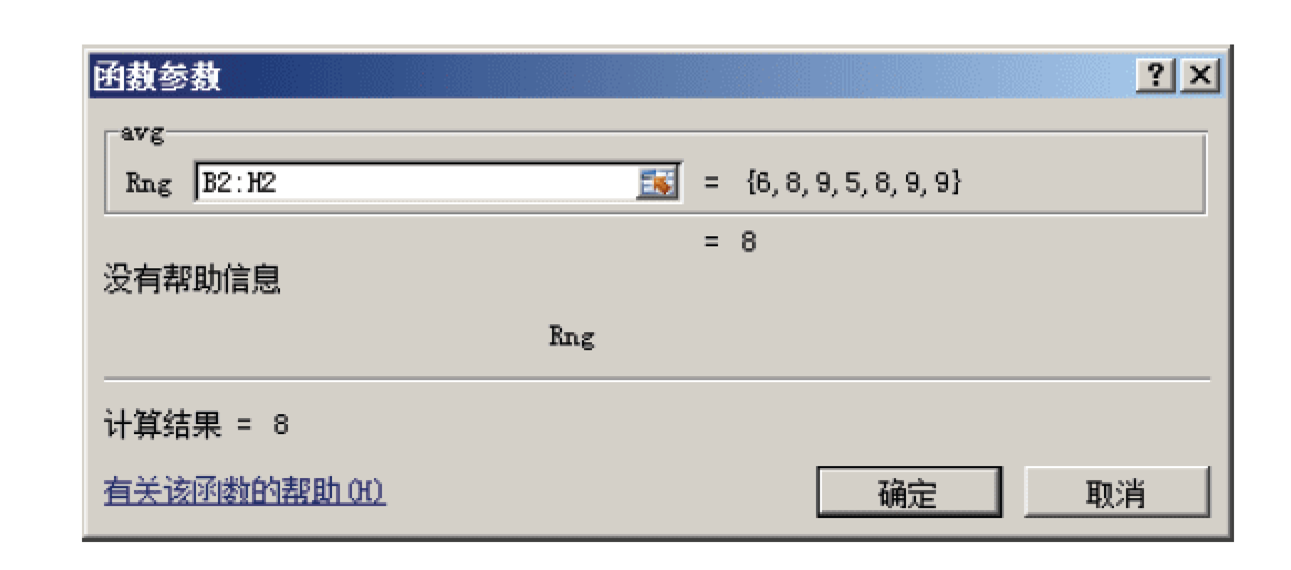


图 8.25　“函数参数”对话框的数据

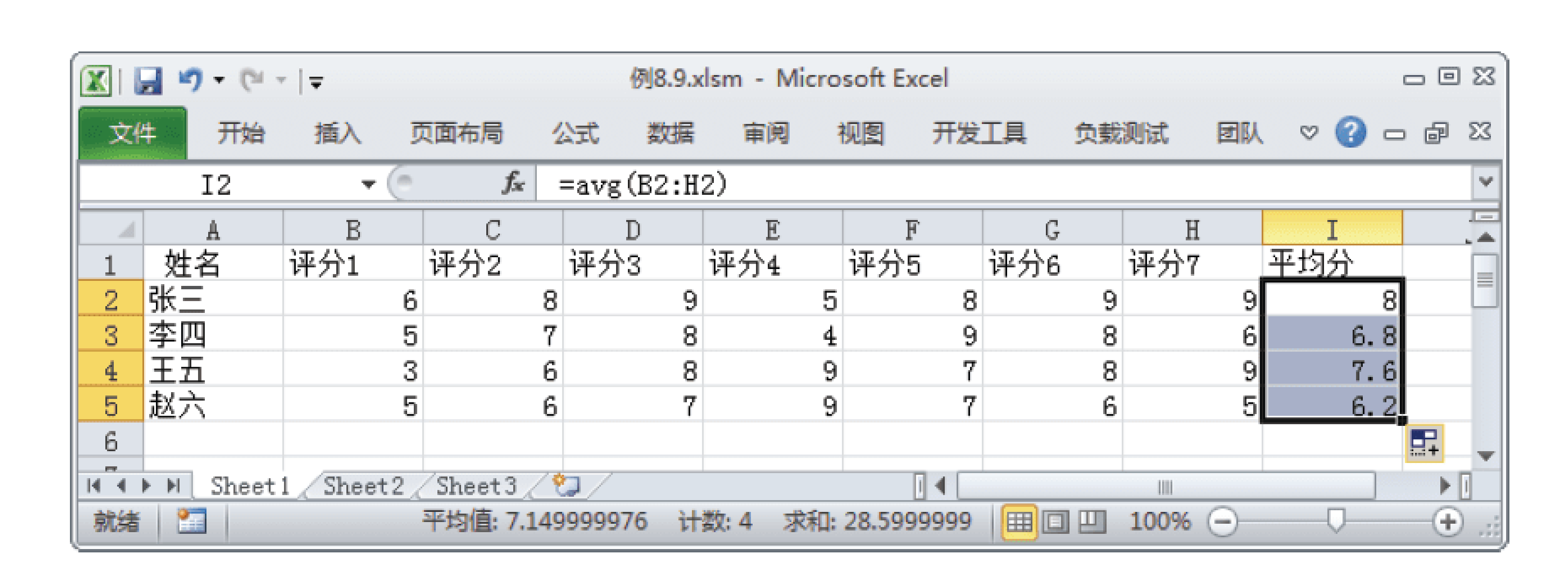


图 8.26　本例的最终效果

8.5　小　　结

本章学习了 VBA 中过程和自定义函数的创建和使用方法。过程是 VBA 程序的基本单元，应用程序实际上就是一个个实现功能的过程的集合。通过本章的学习，读者将掌握过程的创建和调用的方法，以及调用过程时过程间传递参数的方法。VBA 的函数包括 VBA 自带的函数和用户自定义函数。通过本章的学习，读者应该掌握自定义函数的创建以及使用 VBA 程序和在工作表中的不同调用方法，能够熟练应用函数来扩展 Excel 的功能。

Function 函数和 Sub 过程都属于 VBA 的通用过程，它们都是构成程序的基本单位，能够使用 Private 和 Public 等关键字来设定作用域，两者都可以接收参数，并且参数的设置是相同的。但两者也有着许多的不同，Sub 过程是不能返回值的，而 Function 函数可以返回值。Sub 过程可以作为 Excel 的宏来调用，而 Function 函数则不会出现在“宏”列表中，

也就是说不能单独运行。Sub 过程在程序中可以作为单独的语句来调用，而 Function 函数通常是作为表达式的一部分来使用。

8.6　本 章 习 题

1．下面哪个是正确的过程名？（　　）

A．3My_Open　　B．My_Open　　C．My.Open　　D．My Open

2．下面哪个关键字不能用于声明过程？（　）

A．Private　　B．Public　　C．Friend　　D．Optional

3．选择下面程序的运行结果（　　）

```
Sub main()
    Dim a As Integer
    a = 10
    byvali a
    Debug.Print  a
End Sub
Sub byvali(ByVal b As Integer)
    b = b * 10
End Sub
```

A．10　　B．100　　C．5　　D．没有任何输出

4．选择下面程序的运行结果（　　）

```
Sub main()
    Dim a As Double
    calc a, 1, 2, 3, 4, 5
    Debug.Print  a
End Sub
Sub calc(b As Double, ParamArray c())
    Dim n As Integer
    b = 1
    For n = LBound(c) To UBound(c)
        b = b * c(n)
    Next
End Sub
```

A．15　　B．120　　C．100　　D．没有任何输出

5．通过自定义函数在选择区域中查询某个级别产品的个数。

【提示】创建自定义函数，函数用来实现对选定单元格中相同字符的计数。主程序使用 Inputbox 函数来实现需查询字符的输入，调用函数进行查询后将结果使用 MsgBox 函数显示。

6．编写自定义函数，使该函数能够对单元格区域中排位靠前的数据进行求和。

【提示】首先创建自定义函数（代码见附书光盘）。这里，使用 For…Next 循环结果来遍历所选择的单元格，使用 WorksheetFunction 对象的 Large 方法来返回单元格区域的单元格区的第 q 个最大值，如果 q 设置为 4，则为第 4 个最大值。比较单元格的值与这个值的大小，大于等于它的为需要的值，将这些值求和累加即可。

第 2 篇　Excel VBA 编程进阶

⏭ 第 9 章　对象模型

⏭ 第 10 章　Application 对象

⏭ 第 11 章　工作簿对象

⏭ 第 12 章　工作表对象

⏭ 第 13 章　单元格对象

⏭ 第 14 章　工作表界面

⏭ 第 15 章　自定义 Excel 用户窗体

⏭ 第 16 章　自定义 Excel 2010 功能区

⏭ 第 17 章　控制图表

⏭ 第 18 章　类模块

⏭ 第 19 章　数据库编程

第 9 章　对 象 模 型

VBA 是面向对象的编程语言，VBA 编程的本质就是利用 Visual Basic 语句来对 Excel 的对象进行操作。对象本身具有属性、方法和事件；对象像其他变量一样，可以定义对象变量，还可以建立对象数组，增加程序处理的灵活性。Excel 2010 中具有丰富的对象模型，并有一定的层次结构。本章的主要内容和学习目的如下：

- ❑ 理解对象的属性、方法和事件的概念；
- ❑ 掌握对象变量和对象数组的使用方法；
- ❑ 了解 Excel 2010 集合对象的概念；
- ❑ 掌握 Excel 2010 的对象模型和对象层次结构。

9.1　认识 Excel 对象

在 Excel 中，对象指的是一组属性和这组属性上的专用操作封装体。Excel 中包含了很多的对象，工作簿、工作表、单元格、图表、用户窗体以及窗体上的控件等都属于对象。任何一个对象都具有属性、方法和事件这 3 个要素。在建立一个对象后，对对象的操作就是通过对对象的有关属性、事件和方法的操作来实现的。

9.1.1　理解对象的属性

每个对象都有特性，称为对象的属性，其决定了对象所具有的特征，如其大小、颜色和位置等。同时，属性也决定了对象在某一方面的行为，如对象是否可见、是否可以使用等。

在对对象进行操作时，更改对象的属性，就可以更改对象的特性。在 VBA 中，更改对象属性可以通过两种方法来实现，一个是可以直接在对象的“属性”窗口中进行修改，另一种方式是通过程序代码来进行修改。同时，对象的属性值也可以通过程序代码来读取，从而获知对象当前所处的状态。通过代码设置对象的属性值，一般采用下面的语法格式：

```
对象名.属性=值或表达式
```

参数说明如下所示。

- ❑ 对象名：使用对象的名称。
- ❑ 属性：对象的属性名。
- ❑ 值或表达式：赋予对象属性的属性值。

如果需要读取对象的某个属性值，可以采用下面的语法结构：

```
变量=对象名.属性
```

【范例 9-1】 取得工作表窗口宽度和高度值，并重新设置窗口的大小，代码如下所示。

```
01  Sub 设置窗口大小()
02      Dim w As Single, h As Single            '声明变量
03      w = ActiveWindow.Width                  '获得活动窗口宽度
04      h = ActiveWindow.Height                 '获得活动窗口高度
05      MsgBox "当前窗口的宽度为: " & w ,, _
06      & "高度为: " & h & "。", vbOKOnly, "提示"'给出提示
07      ActiveWindow.WindowState = xlNormal     '设置窗口状态
08      ActiveWindow.Width = 300                '设置窗口宽度
09      ActiveWindow.Height = 300               '设置窗口高度
10      ActiveWindow.Caption = "新的数据表"      '设置窗口标题
11     End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码，按 F5 键运行程序，首先提示活动窗口的宽度和高度值，如图 9.1 所示。单击“确定”按钮关闭“提示”对话框后，活动窗口的大小发生改变，窗口标题文字发生改变，如图 9.2 所示。

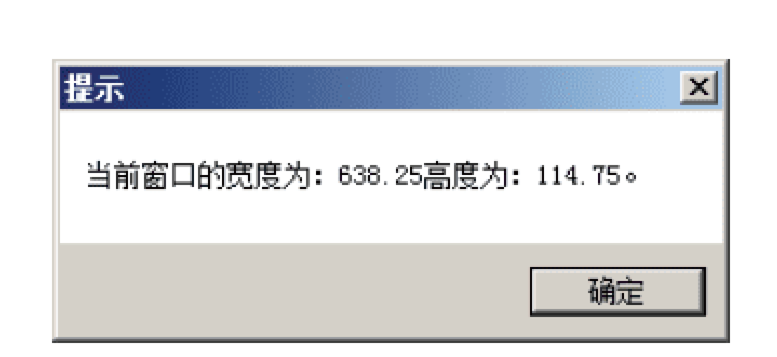
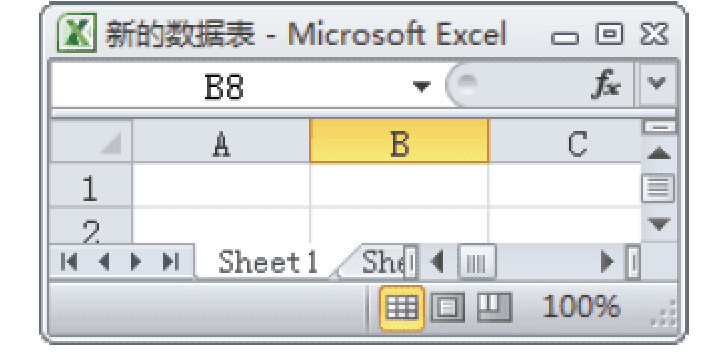


图 9.1　提示当前窗口大小



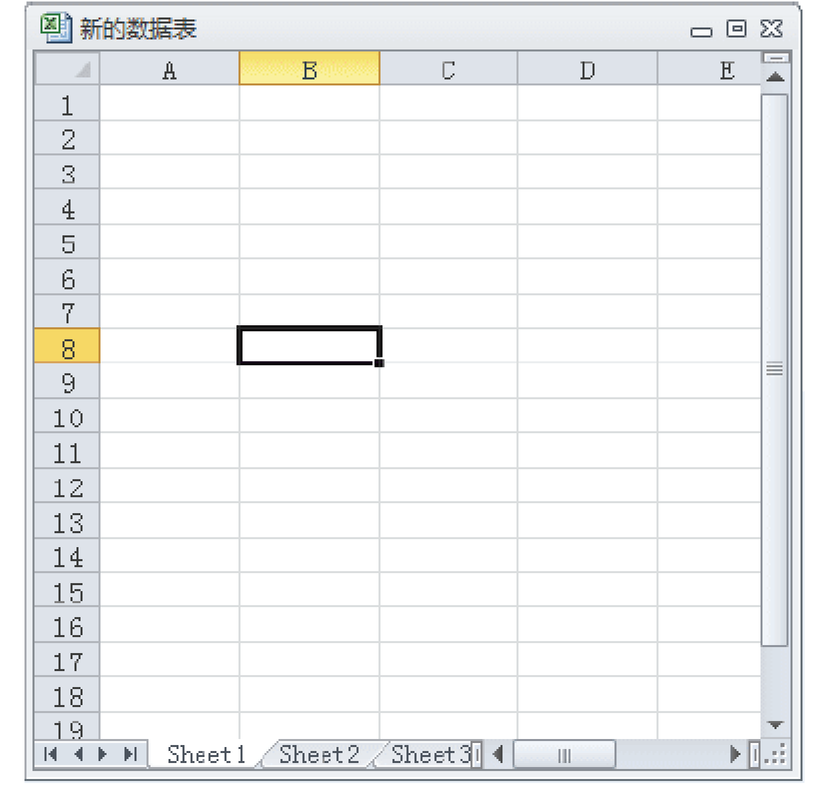


图 9.2　窗口大小和标题改变前后的效果对比

【代码解析】本段用于说明对象属性的意义及更改属性值的方法。代码的第 02 行和第 03 行获取窗口对象的 Width 属性和 Heigth 属性，这两个属性表示活动窗口的宽度和高度值。第 07 行代码中的 WindowState 属性用于返回或设置窗口当前的状态，其可使用的属性值是 3 个，分别是 xlNormal、xlMinized 和 xlMaximized，分别对应正常状态、最小化和最大化。第 08～10 行代码通过更改 Width、Height 和 Caption 属性值来改变窗口的大小和标题。

注意：在前面章节中曾经学习了 With 语句，如果需要设置同一个对象的多个属性，使用该语句将更简单，程序结构更清晰。本例的第 06~07 行就可以使用该语句，读者可以试一试。

9.1.2　理解对象的方法

方法是对象能够执行的动作，它是对象本身包含的函数或过程，用于完成特定的功能。方法可以用来改变对象的属性值，也可以用来对存储在对象中的数据实现某种操作。在

VBA 中，方法是属于对象的，必须通过对象才能对其进行方法。

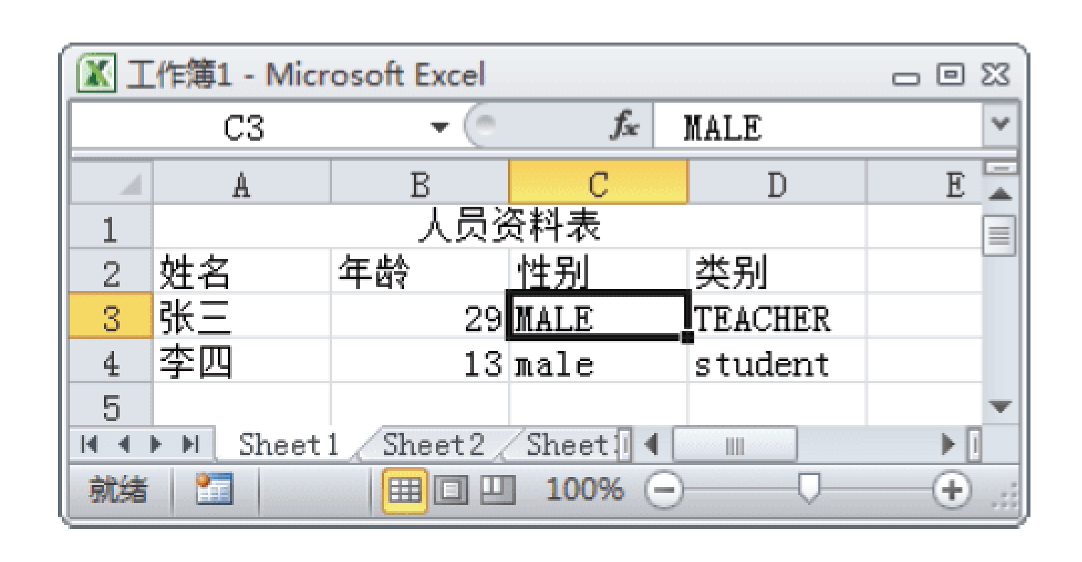
调用对象的方法时，使用点操作符（.）引用方法，如果有参数，在方法后直接加上参数值，参数间使用空格隔开。在程序中使用方法的语法格式如下：

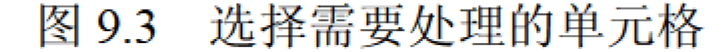
```
对象名.方法
```

【范例 9-2】 使用对象方法将当前选择单元格中英文改为首字母大写其他字母小写的标准形式，代码如下所示。

```
01  Sub 首字母大写()
02      Dim a As String, b As String        '声明变量
03      On Error Resume Next                '错误处理
04      a = Selection.Value                 '获取当前单元格的值
05      b = WorksheetFunction.Proper(a)     '首字母大写
06      Selection.Value = b                 '返回处理结果
07  End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码。在工具表中选择需要处理的单元格，如图 9.3 所示。在 VBA 编辑器中按 F5 键运行程序，程序运行后的结果如图 9.4 所示。




图 9.3　选择需要处理的单元格

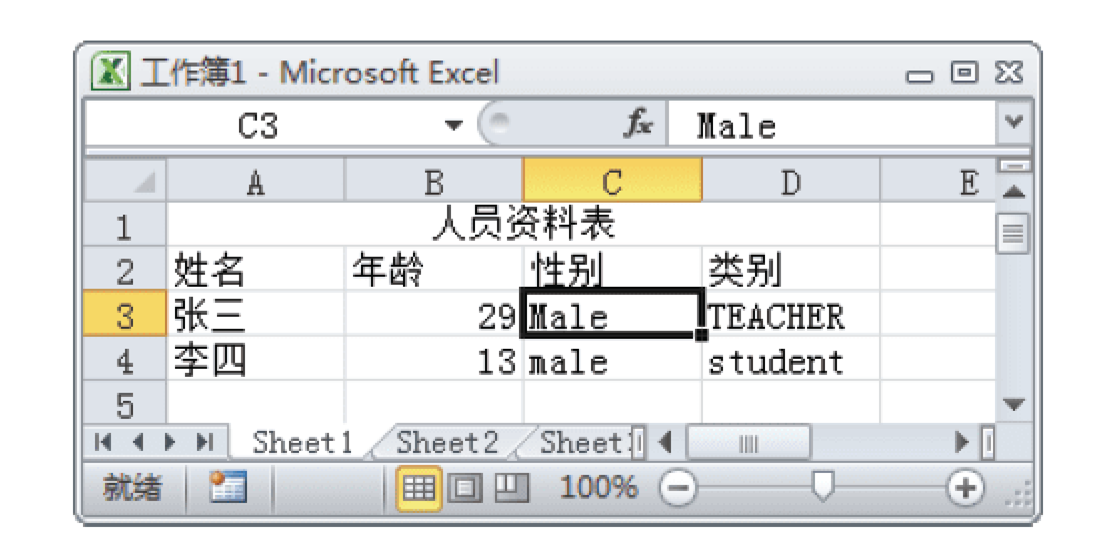


图 9.4　程序运行后的效果

【代码解析】本范例使用 Proper 方法来实现选择单元格中字母格式的转换。代码第 04 行获得选择单元格的内容并将其赋予变量 a，在第 05 行使用 Proper 方法来完成单元格字符格式的转换。完成转换后，通过修改选择单元格对象的 Value 值更改单元格内容。

提示： Proper 方法是 WorksheetFunciton 对象方法，该方法能够将文本字符串的首字母及任何非字母字符之后的首字母转换成大写，同时将其余的字母转换成小写。方法需要参数，参数放在其后的括号中。方法的返回值，使用“=”赋予一个变量。

9.1.3　理解对象的事件

对象事件是在特定时刻发生的事情，是对象状态转换过程的描述，实际上事件就是能够被识别的对象动作。对于这些能够被识别的对象动作，开发者编写相应的代码来对动作做出响应。在 VBA 中，可以激发事件的动作包括：切换工作表、选择单元格以及单击鼠标和打开文档等。当事件发生时，将执行开发者编写的事件响应代码。如果没有事件响应代码，则事件发生将不会有任何的反应。

VBA 的对象事件一般分为两种情况，一种情况是由用户操作所触发的，如鼠标单击、

双击或按键等。另一种情况是由系统或应用程序触发的，如打开工作簿、选定工作表或切换工作表等。

在 VBA 中，对象响应某个事件所执行的操作是通过一段程序来实现的，这段程序称为事件过程。当 VBA 应用程序运行时，会等待一个事件的发生，当事件发生时，程序就会执行事件过程。当完成了这个事件过程后，程序接着进入等待状态，等待下一个事件的发生。如此周而复始直到程序结束。也就是说，事件过程必须由事件触发才能执行，这样的工作模式就是所谓的事件驱动模式。

下面通过一个实例来介绍对象事件代码的编写方法。该实例通过编写工作表的 Activate 事件过程，来实现打开工作表时获得一个提示对话框。

（1）打开 Visual Basic 编辑器并打开“工程”窗口。双击“Microsoft Excel 对象”列表中的“Sheet1”对象打开该对象的“代码”窗口，如图 9.5 所示。

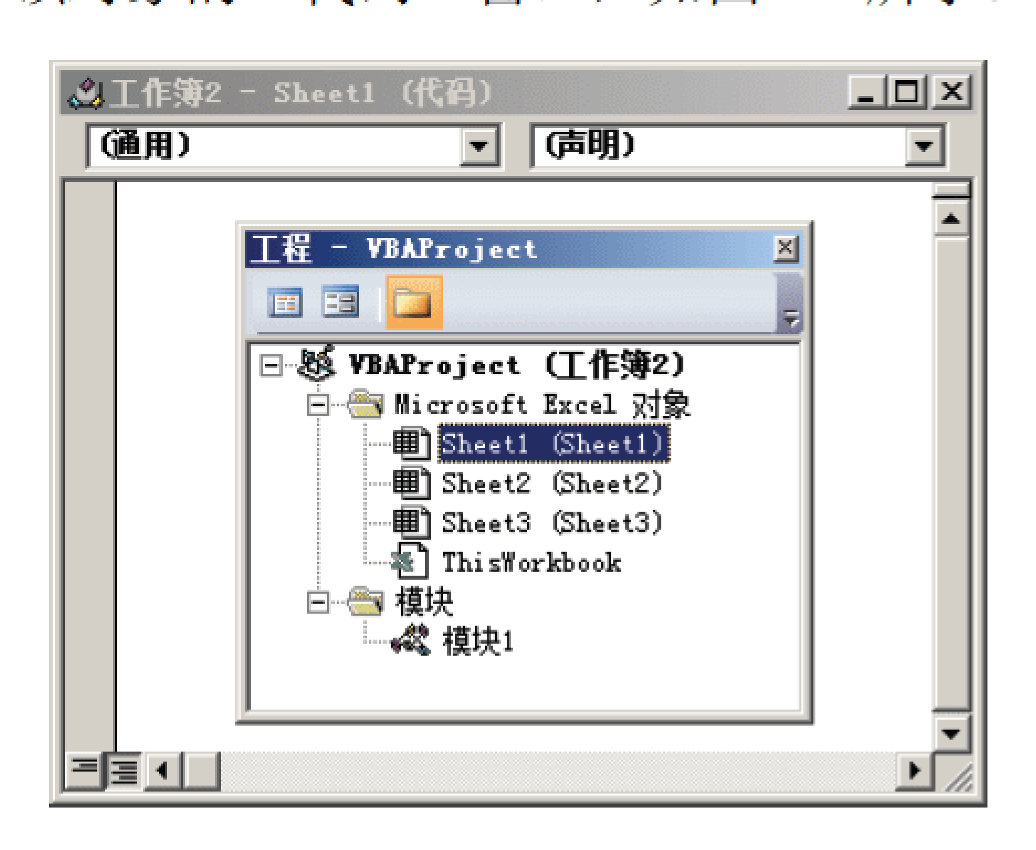


图 9.5　打开“代码”窗口

（2）在“代码”窗口的左上角单击“对象”下拉列表框，在下拉列表中选择需要使用的对象，如图 9.6 所示。在右侧的下拉列表框中选择对象的事件，如图 9.7 所示。



图 9.6　选择对象

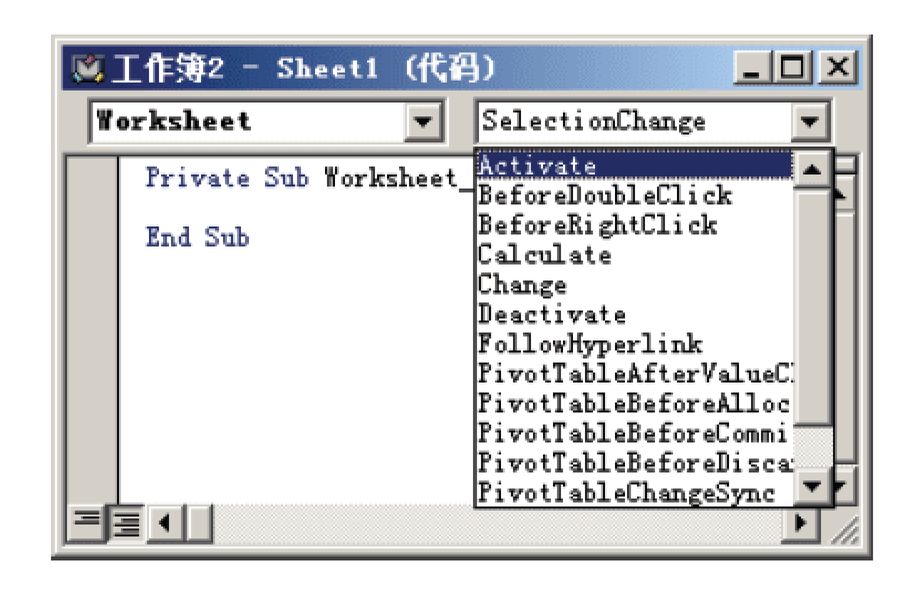


图 9.7　选择事件

注意：这里在创建事件响应时，如果是第一次选择 Worksheet 对象，Visual Basic 会自动使用默认的事件。如果需要使用对象的其他事件，可以在选择需要事件后，将 Visual Basic 自动创建的默认事件过程从“代码”窗口中删除。当然，这里在创建事件过程时，也可以直接在“代码”窗口中输入代码。

（3）此时，Visual Basic 编辑器会自动在代码窗口中添加一个事件响应结构，如图 9.8 所示。在其中输入程序代码即可完成事件编程过程，本实例完整的程序代码如下所示。

```
Private Sub Worksheet activate()                       '使用 activate 事件
    Cells = Clear                                      '清空单元格
    MsgBox "欢迎再次使用此工作表，内容已经被清空，_
        你可以重新输入数据了！", vbOKOnly, "提示"    '给出提示
End Sub
```

（4）完成事件代码的编写后，切换到 Excel 2010。当从其他工作表切换到当前工作表（即 Sheet1）时，发生 Activate 事件，执行事件过程代码。工作表被清空，同时显示提示信息，如图 9.9 所示。

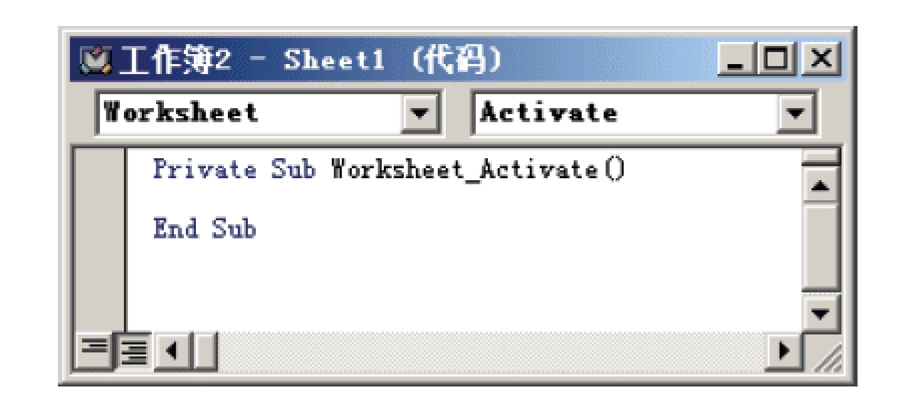


图 9.8　创建基本程序结构图

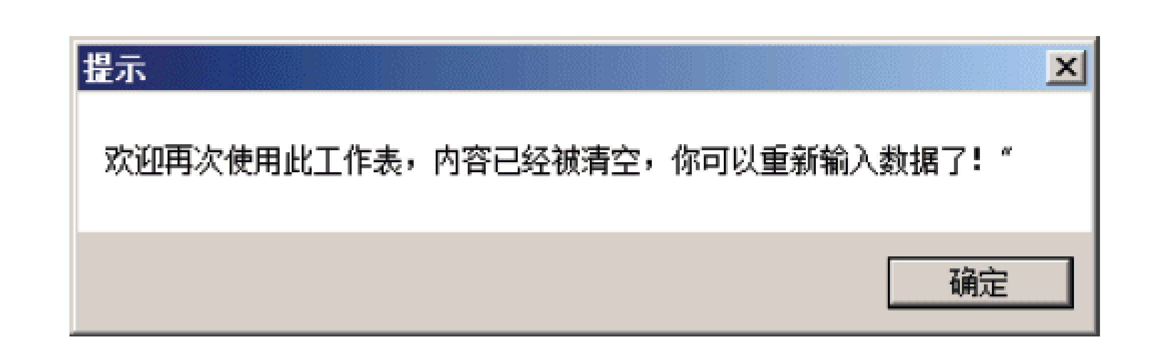


图 9.9　显示提示信息

9.2　使用对象变量和对象数组

在 VBA 中，使用变量和数组同样能够保存或引用对象。通过变量来引用一个在程序反复使用的对象，能够加快程序执行的速度，获得更高的运行效率。

9.2.1　使用对象变量

对象变量是一个能够代表完整对象的变量，其保存着对象引用的指针。在程序中能够方便地实现对对象的引用。在使用对象变量时，一般是先声明对象变量，然后再将变量指定给对象。

对象变量的声明和普通变量的声明相同，使用 Dim、Public、Private 或 Static 等语句来进行声明。引用对象的变量必须是 Variant、Object 或一个对象的指定类型。例如，下面的语句分别将变量 MyObjectA 和 MyObjectB 声明为 Variant 数据类型和 Object 数据类型。

```
Dim MyObjectA As Variant
Dim MyObjectB As Object
```

注意：如果对象变量只有在程序运行时才知道引用的对象类型，可以将其声明为 Object 数据类型。使用 Object 类型可以实现对任意对象的一般引用。如果知道对象变量引用的对象类型，那么就应该声明为这种对象类型，这样能够保证代码生成的速度和代码的可读性。

对象变量同样是可以像普通变量那样进行赋值操作的，但对其赋值必须使用 Set 语句。例如，在完成对 MyObjectA 变量的声明后，下面的语句给对象变量 MyObjectA 赋值：

```
Set MyObjectA= Worksheets(1).Range("A1")
```

在一个过程结束后，应该中断变量与对象的关联以释放内存中占有的空间，此时可以使用 Nothing 关键字来完成。中断变量与对象关联的语句如下所示。

```
Set MyObjectA=Nothing
```

【范例 9-3】 编写程序，将学生成绩表中数学分数高于 90 分的分数以特殊形式显示，代码如下所示。

```
01  Sub 对象变量使用()
02      Dim myCell As Range, n As Integer                       '声明对象变量
03      For n = 3 To ActiveSheet.Range("A1048576").End(xlUp).Row
                                                                '按行遍历单元格
04          Set myCell = Worksheets(1).Cells(n, 3)              '设置对象变量
05          If myCell.Value> 90 Then                            '单元格值如果大于 90
06              myCell.Font.Name = "黑体"                       '设置文字字体
07              myCell.Interior.Color = RGB(255, 200, 214)
                                                                '设置单元格填充颜色
08              myCell.Font.Bold = True                         '设置文字为黑体
09              myCell.Font.Size = 15                           '设置文字大小
10          End If
11      Next
12  End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码，按 F5 键运行程序，程序运行效果如图 9.10 所示。

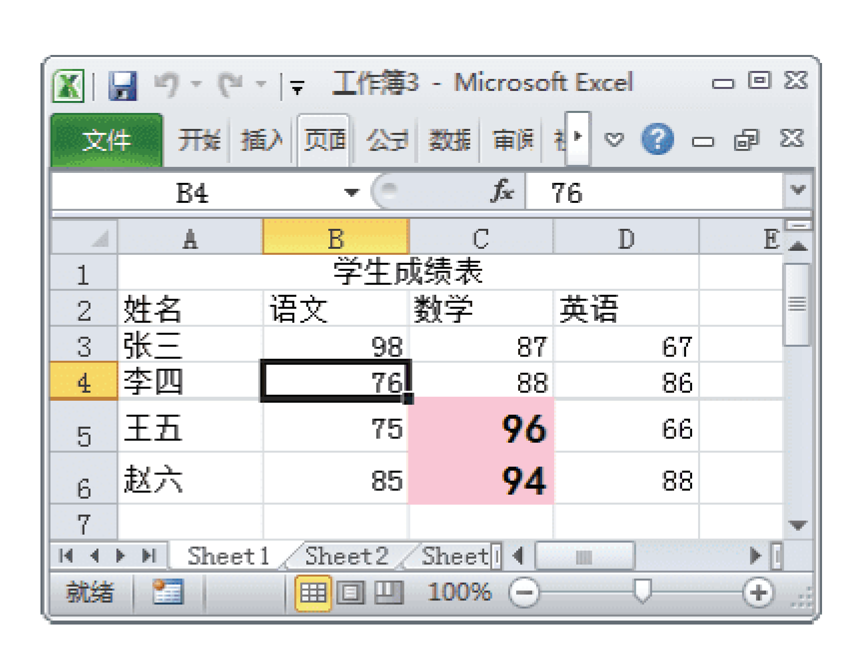


图 9.10　程序运行效果

【代码解析】本示例演示利用对象变量来实现对单元格的操作。在代码中，第 02 行将变量 myCell 声明为 Range 对象。在第 04 行将单元格对象指定给变量 myCell。在程序的第 05~11 行使用变量来引用单元格，对单元格的值、单元格的填充颜色和单元格的文字样式进行设置。

注意： 代码中如果不使用对象变量的话，则需要使用如下的语句：

```
Worksheets(1).Cells(n, n + 1).Font.Name = "黑体"
Worksheets(1).Cells(n, n + 1).Interior.Color = RGB(255, 200, 214)
Worksheets(1).Cells(n, n + 1).Font.Bold = True
Worksheets(1).Cells(n, n + 1).Font.Size = 15
```

提示： 上述代码的可读性显然没有示例中代码那么简洁清晰。同时，如果示例的循环次数很多，那么使用示例中的对象变量将会比上述代码具有更高的代码执行效率。

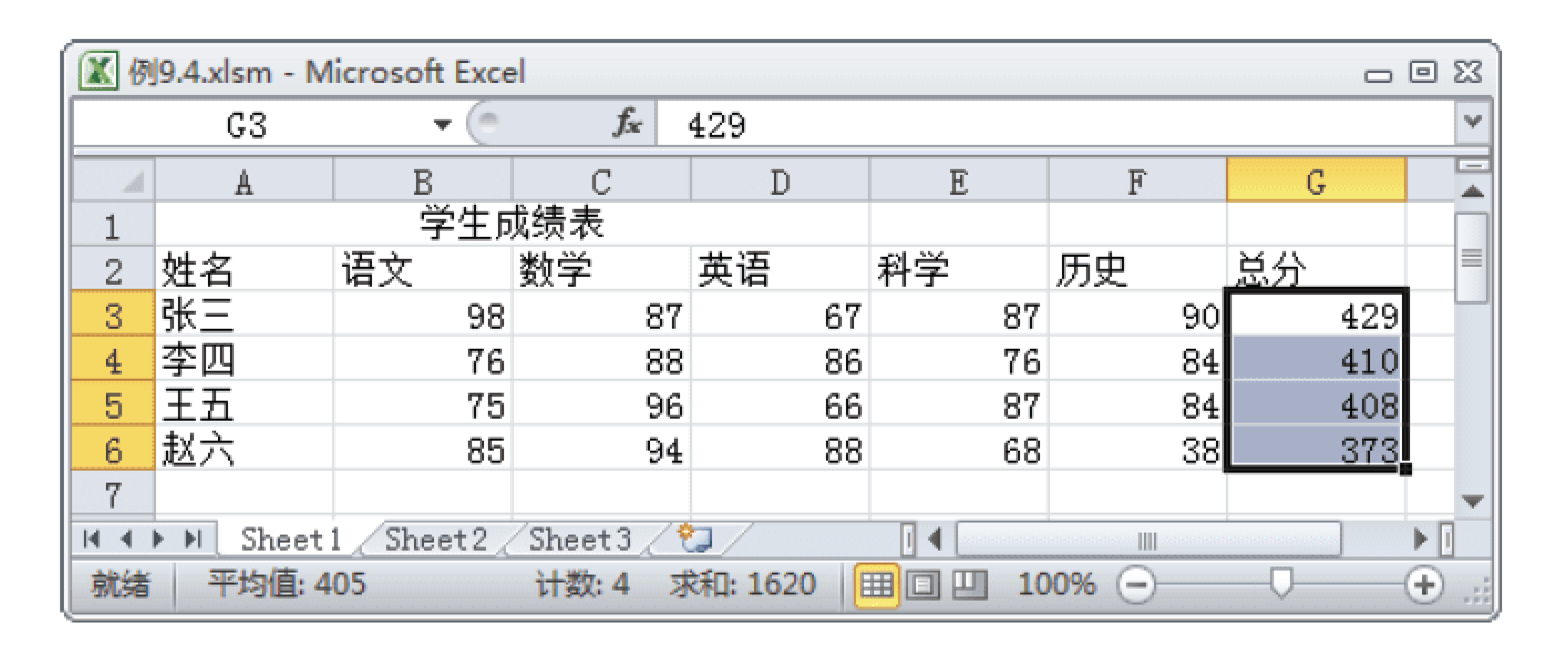
9.2.2　使用对象数组

在程序中如果需要使用大量相同类型的对象，可以使用对象数组来指定这些对象。对象数组的定义和数组的定义方式相同，其使用与对象变量的使用类似。下面通过一个实例来介绍对象数组的使用方法。

【范例 9-4】 使用对象数组来求分数表总分，代码如下所示。

```
01 Sub 对象数组使用()
02     Dim myRange(5, 5) As Range                         '声明对象数组
03     Dim n As Integer, i As Integer, p As Integer, q As Integer
                                                          '声明指针变量
04     For n = 1 To 5
05         For i = 1 To 5
06             Set myRange(n, i) = Worksheets(1).Cells(n + 1, i + 1)
                                                          '为对象数组赋值
07         Next
08     Next
09     For p = 1 To 5
10         For q = 1 To 5
11             Cells(p + 1, 7) = myRange(p, q) + Cells(p + 1, 7)
                                                          '求和结果写入单元格
12         Next
13     Next
14 End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码，按 F5 键运行程序，程序运行效果如图 9.11 所示。



	A	B	C	D	E	F	G
1		学生成绩表					
2	姓名	语文	数学	英语	科学	历史	总分
3	张三	98	87	67	87	90	429
4	李四	76	88	86	76	84	410
5	王五	75	96	66	87	84	408
6	赵六	85	94	88	68	38	373
7							

图 9.11　程序运行效果

【代码解析】示例演示对象数组的使用方法。在这段代码中，第 02 行定义了一个二维数组变量 myRange。代码的第 04～08 行通过 For…Next 循环结构的嵌套来遍历数组变量的所有元素，分别给它们赋值，即将指定行中各个单元格的数据赋予数组。在代码的第 09～13 行将数组中同一行的数据求和后填入指定的单元格中。

9.3　使用 Excel 集合对象

集合实际上是一种特定类型的对象，其能够代表一组相同的对象。在编写应用程序时

使用集合能够优化程序代码，提高程序运行效率。Excel 中的常用集合包括 Sheets 集合、Workbooks 集合、Worksheets 集合以及 Windows 集合。

在 Excel VBA 中，表示某类对象的集合通常使用的是“对象名+后缀 s”的形式（即复数）来表示。例如，WorkSheet 表示工作表对象，而 WorkSheets 表示工作表对象集合。但 Range 是个例外，它可以代表一个单元格，也可以代表某一行、某一列或某个选区，即一个单元格集合。因此，Range 既代表对象又代表集合。在程序中如果需要引用集合中的对象，可以使用下面的语法格式为：

```
集合("对象名" 或 对象索引号)
```

参数说明如下所示。

- 集合：集合的名称。
- 对象名：需要引用集合中对象的名称。
- 对象索引号：需要引用集合中的对象的索引号。

例如，下面代码用于引用 Worksheets 集合中的名为“成绩表”的工作表，可以使用下面两种方法来实现：

```
Worksheets("成绩表")
```

或

```
Worksheets(1)
```

其中后一种方式使用直接索引号来引用工作表，其中的 1 表示其为集合中的一个对象。

如果需要访问集合中的一个对象，可以使用 Item 方法来实现。例如，访问 Worksheets 集合的第一个工作表，并将其赋予对象变量 myWorksheet，可使用下面的语句：

```
myWorksheet=ActiveWorkbook.Worksheets.Item (2)
```

【范例 9-5】 编写程序使用提示对话框显示工作簿中工作表个数，同时删除指定工作表中单元格中的空格，代码如下所示。

```
01  Sub 集合操作示例()
02      Dim numSheet As Long, b As Object                  '声明变量
03      numSheets = ActiveWorkbook.Worksheets.count        '获取集合中对象数量
04      MsgBox "当前工作表有" & numSheets & "个。",
05      vbOKOnly, "提示"                                 '提示工作表个数
06      Set b = ActiveWorkbook.Worksheets.Item(1)        '访问集合中对象
07      b.UsedRange.Replace What:=" ", Replacement: _
08      ="", Lookat:=xlPart                              '删除工作表单元格中空格
09  End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码，按 F5 键运行程序，首先显示提示对话框，提示工作表的个数，如图 9.12 所示。在名为“A 组”的工作表中，单元格存在空格，如图 9.13 所示。关闭提示对话框后，该工作表中的空格被删除，如图 9.14 所示。

【代码解析】本示例演示对象集合的使用方法，通过引用集合中的对象实现对单元格的赋值操作。在代码第 03 行，Count 属性用于获得集合中对象的个数，如果一个集合为空，

则集合的 Count 值为 0。第 06 行代码指定访问的对象，并将其赋予对象变量。在程序的第 07~08 行对指定工作表的单元格进行访问。这里，使用 Replace 方法查找单元格中的空格并将其删除。



图 9.12　“提示”对话框

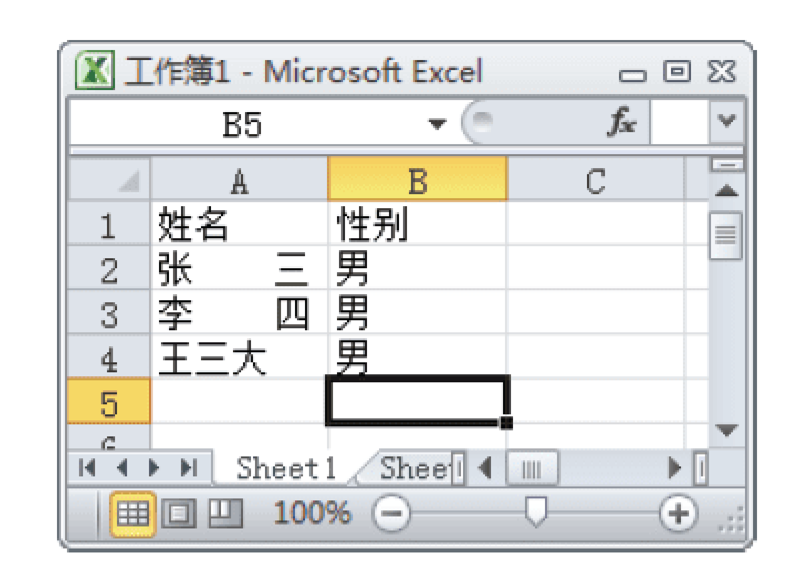


图 9.13　工作表的单元格中存在空格

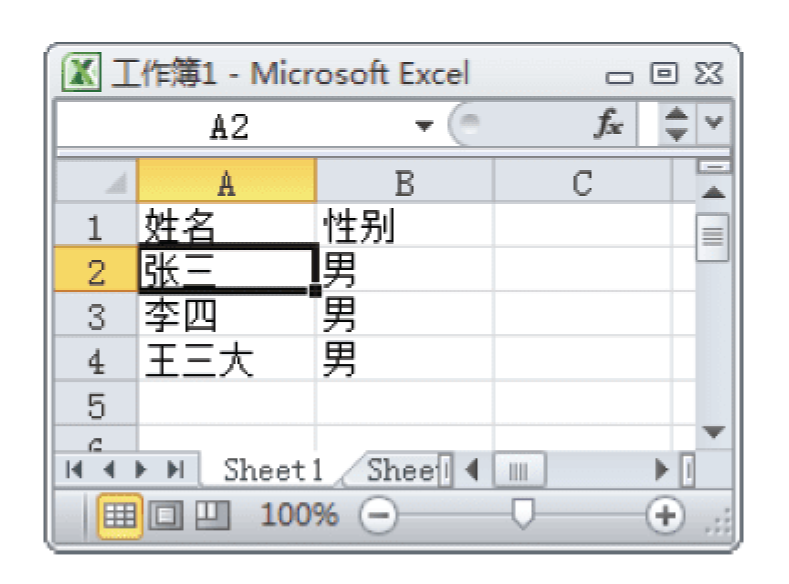


图 9.14　单元格中的空格被删除

注意：Item 方法中的参数是工作表的序号，其并不是工作表默认名称（如 sheet5）中的编号。最直观的确定这个序号的方法就是切换到 Excel，看当前工作簿中需要的工作表按照从左向右排列的位置。如示例中的 Item(4)访问的不是工作表“sheet4”，而是从左向右排在第 4 位的那个工作表，也就是“sheet2”。

9.4　学习 Excel 2010 中的对象模型

对象模型用于描述对象间的关系。Excel 拥有 200 多个对象，对象间如果没有任何关系的话，掌握和使用这些对象是非常困难的。事实上，Excel 中所有的对象都处于一个完整的体系中，每个对象都不是孤立的。

9.4.1　理解 Excel 对象模型

Excel 的对象模型是使用分层管理的方法来实现对 Excel 中的对象的管理。在这种分层模式下，一个对象可以是其他对象的容器，包含有其他对象，同时这些对象又可以被另外的对象所包含。

在 Excel 的对象模型中，位于最顶层的是 Application 对象，代表整个 Excel 应用程序。Application 对象包含 Excel 的其他对象，如 Workbook 对象和 WorkSheet 对象等。由于 Excel 2010 的 VBA 帮助系统中没有使用以前版本的层次结构图，这里为了便于读者的理解，使用了 Excel 2003 对象结构图，从图中可以清晰地看到各个对象间的层次关系，如图 9.15 所示。

提示：Excel 中的对象很多，但对于普通用户来说并不需要掌握 Excel VBA 的所有对象。通常编程的中使用的对象大约只有十多个，读者在学习时抓住主要的对象，如 Application 对象、Workbook 对象、Worksheet 对象、Range 对象和 Chart 对象等，就可以在学习中获得事半功倍的效果。

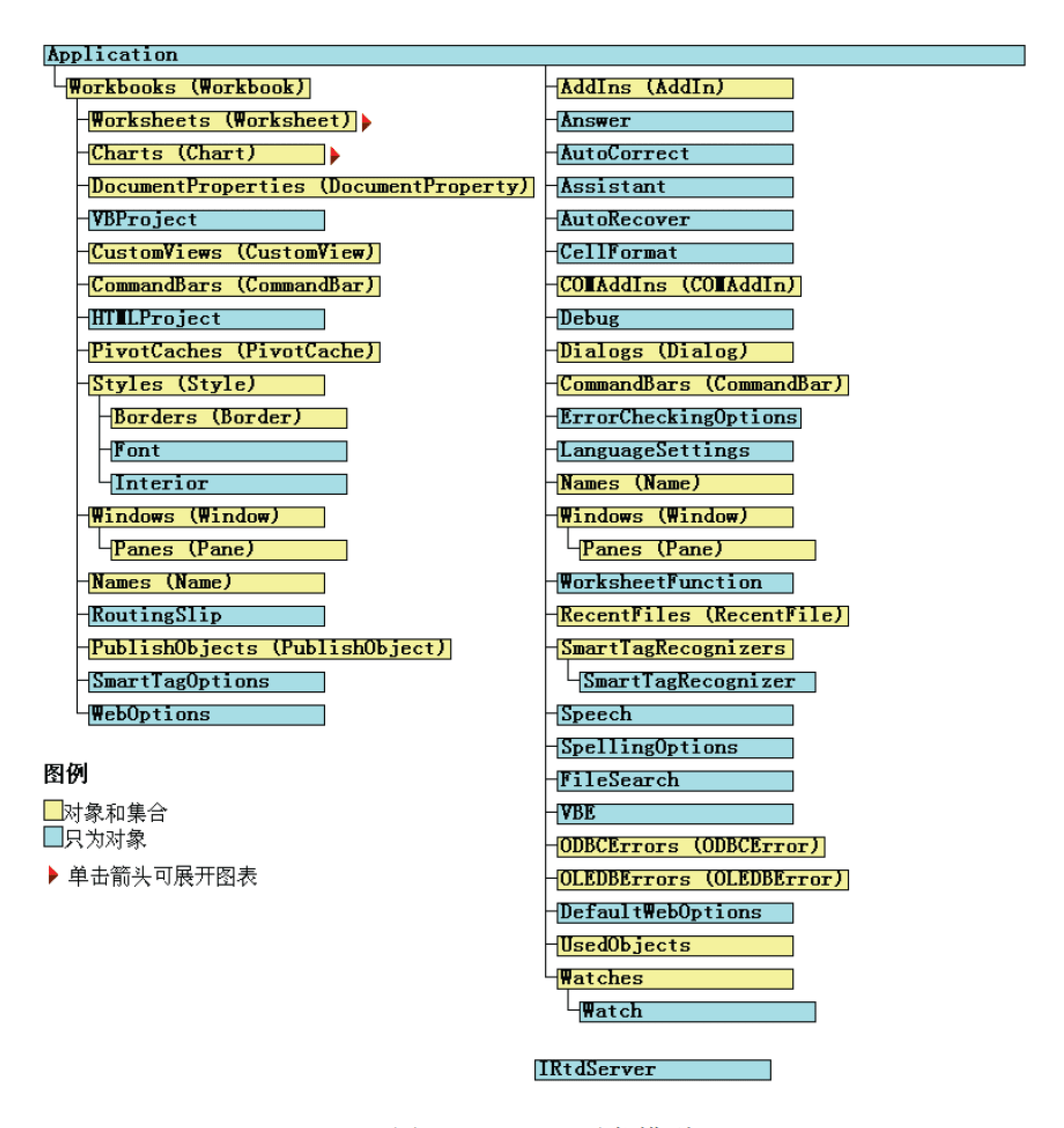


图 9.15　Excel 对象模型

9.4.2　了解 Excel 对象层次结构

掌握了对象的层次结构，在编程中就能够很方便地引用需要的对象。Excel VBA 对象的引用一般采用两种方式，分别是详细引用方式和隐含引用方式。

详细引用方式是一种严密的引用方式。在程序中使用对象，首先应该找到对象在层次结构中的位置，然后再从顶端开始逐项引用对象，直到操作的对象。使用这种方式编程时，需要将对象及其上层对象全部表示出来。如：需要通过 VBA 编程将“职工”工作簿的“工资”工作表的 B3 单元格数据改为 200。要实现这个功能，涉及到工作簿对象、工作表对象和单元格对象，使用详细引用方式的语句如下：

```
Application.Workworkbooks("职工").Worksheets("工资").Range("B3").value=200
```

详细引用在编写代码时十分繁琐，此时可以使用隐含引用方式这种对象引用的简便表示方式。使用此种方式引用对象，是从系统能够确定的与所需对象层次最接近的对象开始引用。如在上面代码中，Application 对象代表的是 Excel 2010 工程，VBA 在 Excel 2010 环境下运行，实际上就隐含了对 Application 对象的引用，因此在代码中隐藏对 Application 对象的引用。这样上述代码可以简写为：

```
Workworkbooks("职工").Worksheets("工资").Range("B3").value=200
```

注意：如果已经打开了“职工”工作簿的“工资”工作表，代码可以简单地写为：

```
Range("B3")=200
```

9.5 小　结

对象是 Excel VBA 编程的重要概念，本章学习了对象的属性、方法和事件的概念，介绍了对象变量和对象数组在程序设计中的作用，同时介绍了集合的概念和 Excel 的对象模型结构。通过本章的学习，读者将对 Excel 2010 VBA 对象的概念有了更深入的了解，为后面深入学习 Excel 2010 VBA 的对象编程打下了深厚的基础。

VBA 是面向对象的可视化编程，程序的编写面向的是可视化的对象。面向对象的程序设计的一个重要工作就是为每个对象设置属性。而面向对象的编程是以对象事件为中心。Excel VBA 程序就是包含若干通用过程和事件过程的集合体。事件触发事件过程，通用过程通过事件触发，也可以通过调用触发。各个事件的发生顺序是不确定的，这就是事件驱动机制。VBA 是面向对象的事件驱动的编程机制，这样的机制能够使程序的设计更为简单，开发者只需要针对过程和事件进行编码就可以了。从第 9 章开始，将学习 VBA 的常用对象及其面向这些对象的编程方法。

9.6 本 章 习 题

1．下面哪种格式能够设置对象属性？（　　）

A．对象名.属性=变量　　B．变量=对象名.属性

C．对象名：属性=变量　　D．对象名_属性=变量

2．当选择工作表中选择单元格的分数 74 时，运行程序，单元格 Cells（2,4）中显示结果是什么？（　　）

```
Sub test ()
    Dim f As Integer
    f =Selection
    Select Case f
        Case 90 To 100
        ActiveWorkbook.Worksheets.Cells(2,4)="优秀"
        Case 70 To 90
        ActiveWorkbook.Worksheets.Cells(2,4)="良好"
        Case 60 To 70
        ActiveWorkbook.Worksheets.Cells(2,4)="及格"
        Case Else
        ActiveWorkbook.Worksheets.Cells(2,4)="不及格"
    End Select
End Sub
```

A．优秀　　B．良好　　C．及格　　D．不及格

3．运行下面程序，单元格中文字是什么颜色（　　）

```
Sub Test ()
    Dim c As Object
     Set c=Selection.Font
     c.Size=16
     c.Color=vbRed
     c.FontStlye="Bold Italic"
End Sub
```

A．褐色　　　　B．蓝色　　　　C．红色　　　　D．绿色

4．编程记录当前工作表的名称，并将其分别写入每个工作表的第一个单元格中。

【提示】在编程时，使用循环结构来遍历工作簿中单元格，循环条件是工作簿中工作表的个数，使用 Count 属性来获得。在循环体中可以使用 Item 方法来指定需要访问的工作表，将工作表名写入该工作表的单元格即可。

5．编写程序设置选择单元格中英文字母的样式。要求字母全部变为大写，并且重新设置其字体、大小和颜色等。

【提示】本练习需要对选择的每个单元格执行操作，使用 For Each…In Next 结构来实现循环。使用该循环结构能够实现集合中对象的遍历。同时，由于需要对集合中相同的对象（即单元格）进行相同的操作，使用 With…End With 结构是一个好的方法。

第 10 章　Application 对象

Excel 提供了多达 200 多种的对象，其中 Application 对象是其中的一种常用对象。Application 对象代表整个 Excel 应用程序，通过使用 Application 对象可以控制程序的运行，可以设置 Excel 软件和界面外观，以及操作单元格和控制 Excel 文件的打开、存储和获取文件信息。本章将介绍 VBA 中 Application 对象的使用方法。本章的主要内容和学习目的如下：

- 掌握使用 Application 对象操作 Excel 文件；
- 掌握通过 Application 对象设置 Excel 的界面外观；
- 掌握通过 Application 对象操作单元格的方法；
- 掌握通过 Application 对象获取、打开或保存 Excel 文件的方法。

10.1　常 用 操 作

使用 Application 对象的方法和属性，能够实现一些常见的操作，如打开文档、实现计时和退出 Excel 等，本节将介绍使用 Application 对象来对 Excel 进行操作的方法和技巧。

10.1.1　在 Excel 中使用“打开”对话框

在进行数据处理时往往需要打开指定的工作簿文档，在 Excel 中可以通过“打开”对话框来完成寻找某个工作簿文档并将其打开的操作。在 VBA 中，使用 Application 对象的 FindFile 方法能够打开一个“打开”对话框以实现 Excel 文档的打开操作。下面通过一个范例来介绍 FindFile 方法的使用技巧。

【范例 10-1】 打开“打开”对话框，代码如下所示。

```
Sub 打开【打开】对话框()
    Dim a As Boolean
    Application.FindFile                            '打开“打开”对话框
    a = Application.FindFile                        '获得返回值
    If a = True Then                                '判断文件是否打开成功
        MsgBox "Excel 文件打开成功！", vbOKOnly, "提示"
                                                    '打开成功，显示提示
    Else
        MsgBox "Excel 文件打开失败！", vbOKOnly, "提示"
                                                    '打开失败，显示提示
    End If
End Sub
```

【运行结果】在事件管理器中创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序将打开“打开”对话框，如图 10.1 所示。当文件打开成功时，显示“提示”对话框，如图 10.2 所示。如果单击“取消”按钮退出“打开”对话框，将显示退出提示，如图 10.3 所示。

【代码解析】本段示例代码介绍了使用 FindFile 方法进行打开文档操作的方法。代码的第 03 行使用 Application 对象的 FindFile 方法打开“打开”对话框。在使用 FindFile 方法时，可以通过判断该方法的返回值来判断打开是否成功，如果打开“打开”对话框成功，将返回 True。如果用户取消该对话框，则返回值为 False。具体的代码编写方法可参考示例的第 05～09 行。

提示： 在“打开”对话框中选择需要打开的 Excel 文档后，单击“确定”按钮关闭对话框，选择的文档将能够在 Excel 中被打开。

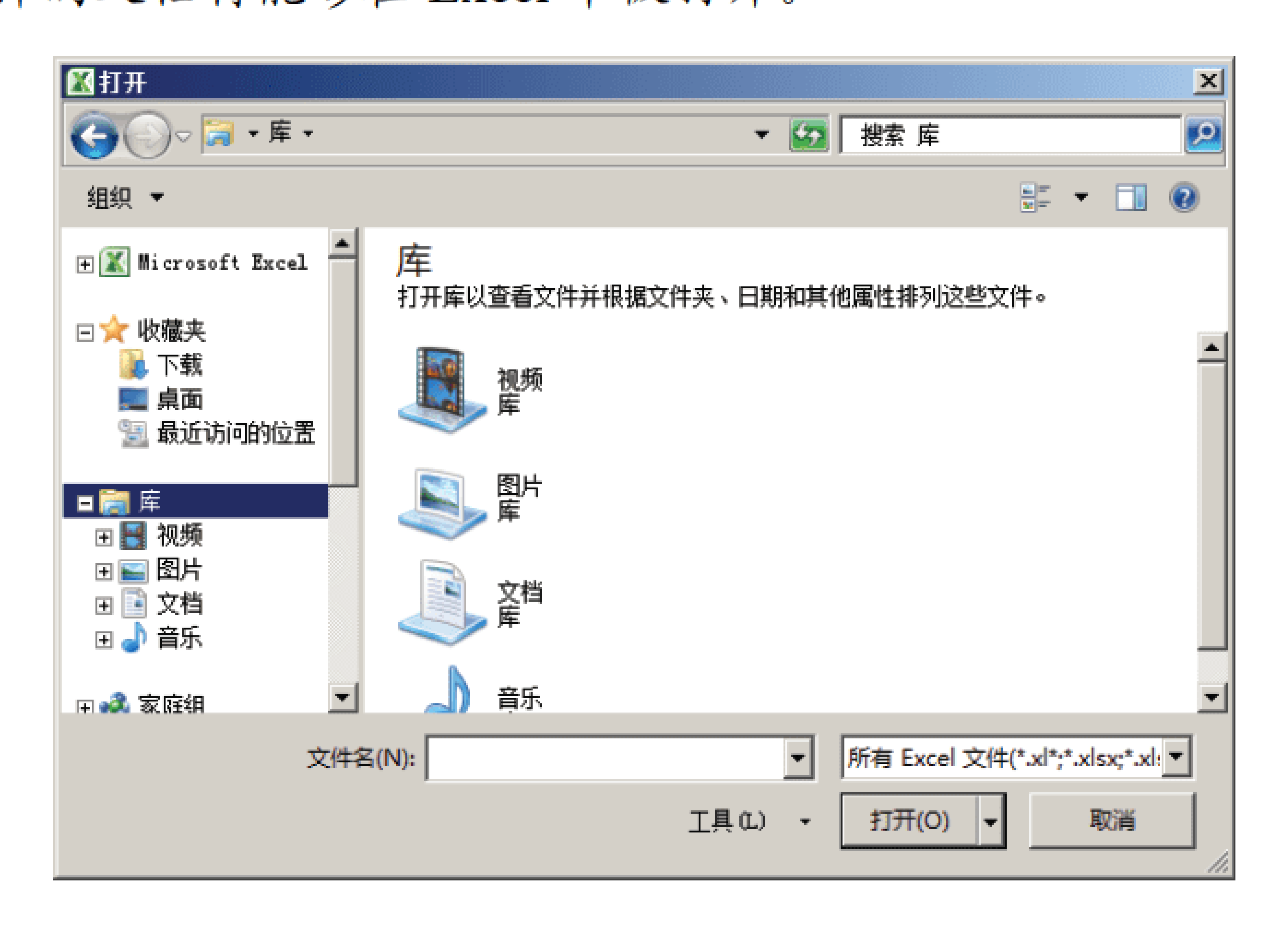


图 10.1　显示窗口大小






图 10.2　文件打开成功的提示对话框

图 10.3　文件打开失败的提示对话框

10.1.2　实现 Excel 定时操作

使用 Application 对象的 OnTime 方法，能够安排一个过程在特定的时间运行。这里的时间，可以是某个具体的时间点，也可以是从程序开始运行时的某个时间段。如果使用该语句安排经过一段时间（从现在开始计时）之后运行某个过程，其语法格式如下：

```
Now + TimeValue(time),
```

在上述代码中，time 为延时的时间，其格式为“时:分:秒”。如果需要指定在某个时

间点运行过程，例如，需要在下午 18 点执行 my_Procedure 进程，可使用下面语句。

```
Application.OnTime TimeValue("18:00:00"), "my_Procedure"
```

如果需要取消对时间的指定，可以使用下面语句。

```
Application.OnTime EarliestTime:=TimeValue("18:00:00"),Procedure:=
"my Procedure",
 Schedule:=False
```

【范例 10-2】 在程序运行 30 秒后显示提示对话框，由用户选择是退出程序还是继续运行，代码如下所示。

```
01  Sub main()
02      Application.OnTime Now + TimeValue("00:00:30"), "my_Procedure"
                                          '设置过程启动时间
03  End Sub
04  Sub my_procedure()
05      a = MsgBox("现在已完成 30" 计时！", vbOKOnly, "提示")      '显示提示
06  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行 main 过程，延时 30 秒后，显示提示对话框，如图 10.4 所示。



图 10.4　提示计时结束

【代码解析】本段程序代码演示使用 OnTime 方法来实现定时操作的方法。本段代码在第 02 行指定经过 30 秒后执行过程“my_Procedure”。同时在代码中创建了一个“my_Procedure”过程，该过程使用 MsgBox 函数给出一个提示对话框，提示时间到。

警告： 这里，在指定需要运行的过程时，过程名必须使用字符串数据，这也就是为什么过程名用引号括起来的原因。

10.1.3　退出 Excel 应用程序

使用 Application 对象的 Quit 方法能够退出当前的 Excel 程序。使用此方法时，如果未保存的工作簿处于打开状态，则 Excel 将显示一个对话框，询问是否要保存对文档的更改。如果文档已经保存，则直接退出。

如果要避免在使用 Quit 方法时出现 Excel 的保存文档提示对话框，可以在使用 Quit 方法前先保存所有工作簿。另外，也可以将 DisplayAlerts 属性设置为 False。这样，在 Excel 退出时，即使工作簿未保存，也不会显示保存对话框，而且 Excel 将不保存文档直接退出。

【范例 10-3】 运行程序，程序给出提示对话框由用户选择是否退出 Excel，代码如下所示。

```
Sub 退出 Excel()
    a = MsgBox("真的退出 Excel 吗？", vbOKCancel, "提示")  '显示提示对话框
    If a = vbOK Then                                   '如果单击了“确定”按钮
        Application.Quit                               '退出 Excel
```

```
    Else
        Exit Sub                                  '退出过程不退出 Excel
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码窗口”输入以上代码，按 F5 键运行程序，程序给出提示对话框，如图 10.5 所示。单击“确定”按钮，将退出 Excel。如果单击“取消”按钮，Excel 将不会关闭。在退出 Excel 时，如果文档没有保存，Excel 会给出提示对话框提示是否保存文档，如图 10.6 所示。

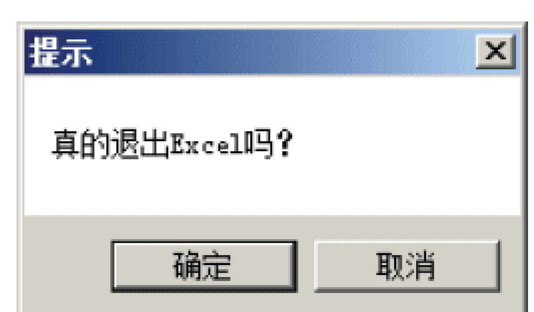


图 10.5　程序给出提示对话框



图 10.6　Excel 提示是否保存文档

【代码解析】本段代码模拟退出 Excel 的操作过程。程序运行时，将首先给出提示对话框由用户做出选择。在程序中使用 If 结构判断用户做出的选择。当选择退出时，使用 Quit 方法关闭 Excel 应用程序，否则将只退出进程而不退出 Excel。

提示： vbOk 是一个常量，是单击提示对话框的“确定”按钮时 MsgBox 函数的返回值。对于 MsgBox 函数来说，其返回值还包括 vbCancle、vbIgnore、vbYes、vbNo 和 vbRetry 等，其分别代表按下“取消”按钮、“忽略”按钮、“是”按钮、“否”按钮和“重试”按钮等。

10.1.4　在 Excel 过程中调用宏

在编写基于 Excel 的应用程序时，有时需要调用录制的宏。在 VBA 中，可以使用 Application 对象的 Run 方法来运行指定的宏。下面通过一个范例来介绍在 VBA 程序中运行宏的方法。

【范例 10-4】 打开范例 10-2 创建的文档，使用 Run 方法调用名为“main”的宏（即 main 过程），代码如下所示。

```
01  Sub 调用宏()
02      Application.Run "main"                          '调用宏
03  End Sub
```

【运行结果】打开文档的 Visual Basic 编辑器后，插入一个新模块，在模块的“代码”窗口输入以上代码，按 F5 键运行程序，程序调用宏后会给出与范例 10-2 相同的提示对话框，如图 10.7 所示。



图 10.7　程序给出提示

【代码解析】代码使用 Run 方法来调用文档中存在的宏，该方法的语法格式如范例第 02 行所示。使用该方法除了可以运行宏外，还可以运行 Dll（动态链接库）或 Xll 中的函数。

10.1.5　激活 Office 2010 应用程序

在 VBA 中，可以使用 Application 对象的 ActivateMicrosoftApp 方法来启动和激活 Microsoft 应用程序。该方法在使用时需要指明需激活的程序，用于指定激活程序的常量包括 7 个：xlMicrosoftWord、xlMicrosoftPowerPoint、xlMicrosoftMail、xlMicrosoftAccess、xlMicrosoftFoxPro、xlMicrosoftProject、xlMicrosoftSchedulePlus，这些常量分别用来激活 Word、PowerPoint 以及 Access 等 Microsoft 应用程序。

【范例 10-5】 通过编写 VBA 程序代码调用 PowerPoint，代码如下所示。

```
Sub 调用 Microsoft 程序()
    Application.ActivateMicrosoftApp xlMicrosoftPowerPoint
                                                    '调用 PowerPoint
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，如果 PowerPoint 已经启动，则切换到该程序。如果 PowerPoint 未启动，则启动 PowerPoint，如图 10.8 所示。

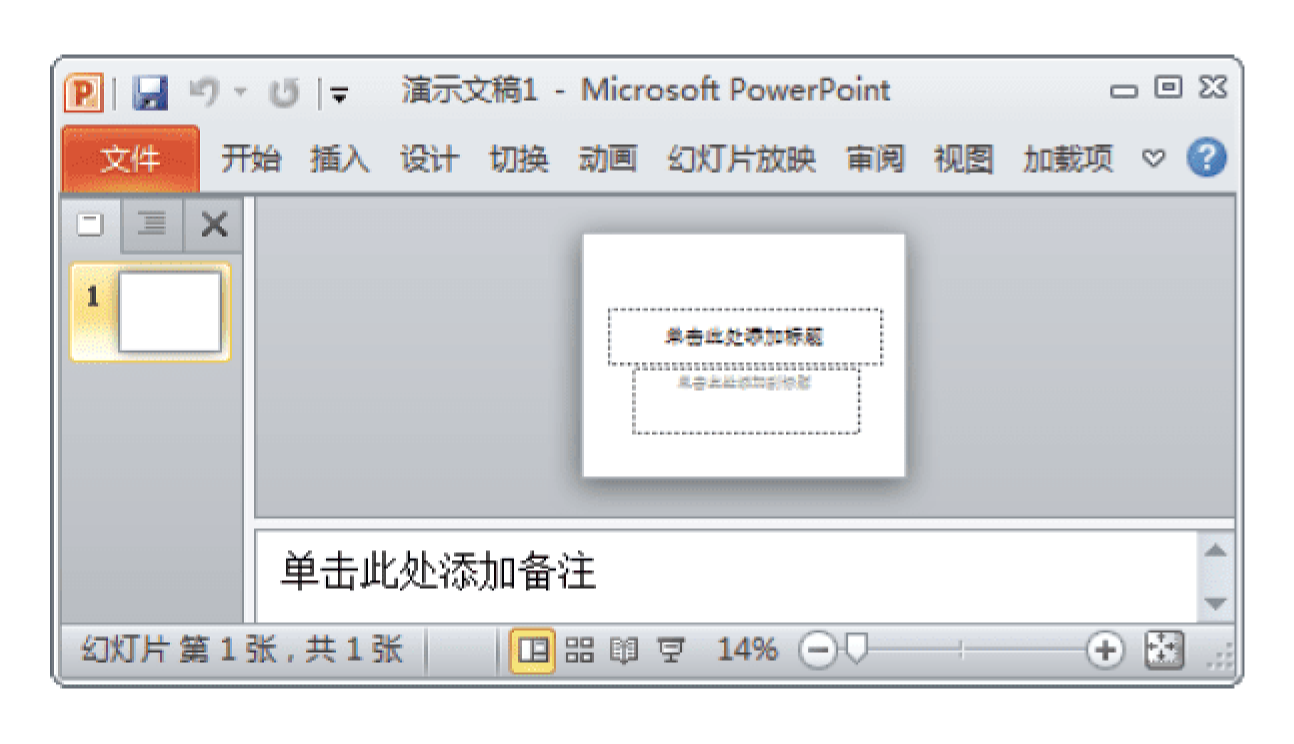


图 10.8　启动 PowerPoint

【代码解析】在上述程序中使用了 Application 对象的 ActivateMicrosoftApp 来激活 Microsoft 应用程序，使用 xlMicrosoftPowerPoint 常量将需激活的程序指定为 PowerPoint 2010。

10.1.6　获取 Excel 系统信息

这里提到的系统信息包括 Excel 版本信息、用户信息和操作系统信息。Excel 为用户能够查阅这些信息提供了专门的属于 Application 对象的属性。

Application 对象的 CalculationVersion 可以返回一个版本数字，数组最右侧两位表示计算引擎的次要版本号，而左侧的其他位表示 Microsoft Office Spreadsheet 组件的主要版本号。

在程序中，使用 OperatingSystem 属性可以获得当前操作系统的名称和版本号，该属性值为 String 数据类型，而且是只读的。

Application 对象的 OrganizationName 的属性值包含注册组织名称，该属性值是 String 数据类型，并且是只读的。如果需要获得当前用户的用户名，可以读取 UserName 属性值，该属性可以返回或设置当前用户的名称。

【范例 10-6】 编写程序查阅系统信息，代码如下所示。

```
Sub 查阅系统消息()
   MsgBox "Excel 版本信息为:" & Application.CalculationVersion & Chr(13) _
   & "本机操作系统的名称和版本为:" & Application.OperatingSystem & Chr(13) _
   & "本产品所登记的组织名为:" & Application.OrganizationName & Chr(13)
   & "当前用户名为:" & Application.UserName & Chr(13) _
   & "当前使用的 Excel 版本为:" & Application.Version, vbOKOnly, "系统信息"
     '显示系统信息
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序给出记录系统信息的对话框，如图 10.9 所示。

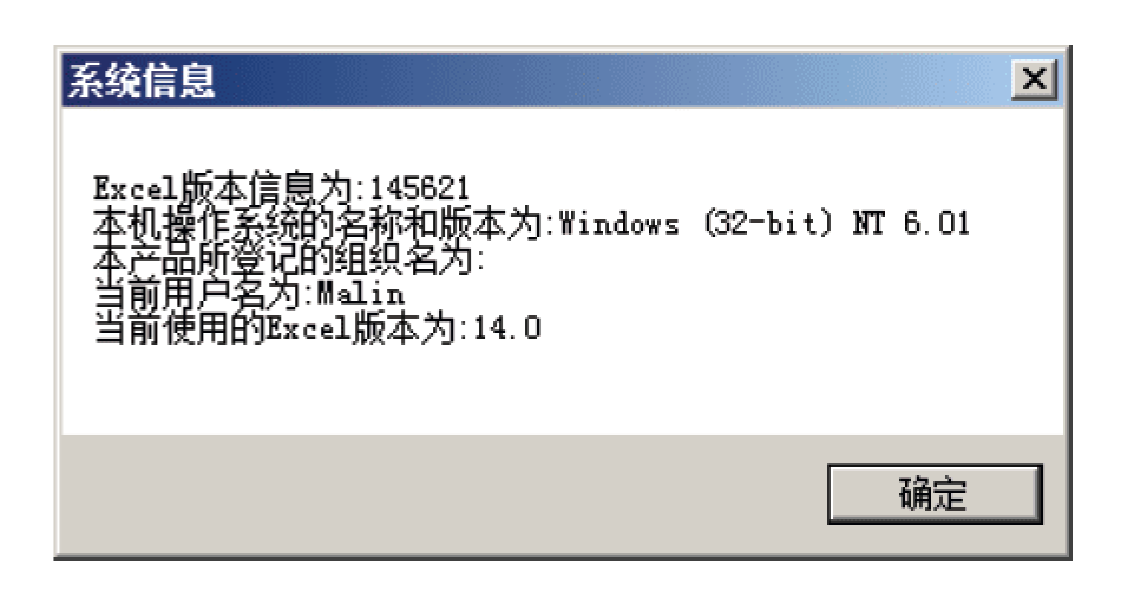


图 10.9　显示系统信息

【代码解析】代码使用 MsgBox 函数来显示有关系统信息，信息通过调用本节介绍的几个 Application 对象的属性值获得。在程序中使用 Chr（13）语句来对实现文字的换行。由于文字比较多，为了便于查看，对代码进行了断行处理。

10.1.7　为 Excel 操作指定快捷键

在 Excel 中，可以给某个过程的启动指定快捷键，这样可以在 Excel 窗口中直接使用快捷键运行该过程代码。为某个过程指定快捷键，可以使用 OnKey 方法来实现。下面通过一个范例来介绍使用 OnKey 方法指定和取消快捷键的方法。

【范例 10-7】 将“Ctrl+Shift+Q”键指定为启动程序的快捷键，代码如下所示。

```
Sub 指定快捷键()
   Application.OnKey "+^{q}", "myPro"                '指定快捷键
End Sub
Sub myPro()
   MsgBox "当前快捷键为 Ctrl+Shift+Q，关闭此对话框后快捷键将取消。"
   Application.OnKey "+^{q}"                         '取消快捷键的指定
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行“指定快捷键”过程指定快捷键。在 Excel 窗口中按 Ctrl+Shift+Q 键即可运行“myPro”过程，

该过程代码运行后显示的提示信息如图 10.10 所示。关闭提示对话框后，快捷键取消。



图 10.10　显示提示信息

【代码解析】在代码中，“指定快捷键”过程用于使用 OnKey 方法为 myPro 过程的启动指定快捷键。myPro 过程在运行时提示快捷键已经指定，在关闭提示对话框后将指定的快捷键取消。代码的第 02 行使用 OnKey 方法指定的是一个组合键，其中“+”表示“Shift”键，“^”表示“Ctrl”键，而“Alt”键用“%”表示。在指定快捷键时，字母键用对应字母表示，功能键使用其名称，如回车键为 Enter，空格键为 BackSpace，向右的方向键为 Right 等。当需要取消快捷键的指定时，可像第 06 行代码那样，将过程指定为空即可。

注意：在指定快捷键时，一般应该将键盘上的功能键（如“F2”，“F8”）等或其他在应用程序中用的比较少的键分配给程序。另外，在退出程序时，应该取消快捷键的指定，这些都是为了避免与其他程序的快捷键冲突。

10.2　设置 Excel 界面外观

在编写基于 Excel 的应用程序时，有时需要对 Excel 操作界面的外观进行修改，以满足不同应用程序的需要。通过修改 Application 提供的界面属性，可以对 Excel 的界面外观进行控制，如设置应用程序标题、状态栏文字和鼠标形状等。

10.2.1　设置 Excel 界面标题栏

Excel 程序窗口的标题栏在默认状态下将显示当前文档的文档名，在创建自己的应用程序时，往往需要根据需要更改标题栏文字。在 VBA 中，使用 Application 对象的 Caption 属性能够改变 Excel 主窗口标题栏显示的名称。如果将代码放置在工作簿的 Open 事件代码段中，就可以实现启动 Excel 时自动更改标题栏名称。

【范例 10-8】 更改状态栏显示的文字，代码如下所示。

```
Sub 修改标题栏()
    Dim myString As String, a As String
    a = Application.Caption                          '获得当前标题栏名称
    myString = InputBox("当前标题栏显示的是: " & " “" & a & "”, _
    " & "请输入新的标题栏名称", "提示")                 '显示输入对话框
     Application.Caption = myString                   '修改标题栏名称
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序首先打开输入对话框，提示当前主窗口标题，并要求输入新的标题栏名称，如图 10.11 所示。完成输入后关闭该对话框，Excel 标题栏名称改为需要的文字，如图 10.12 所示。

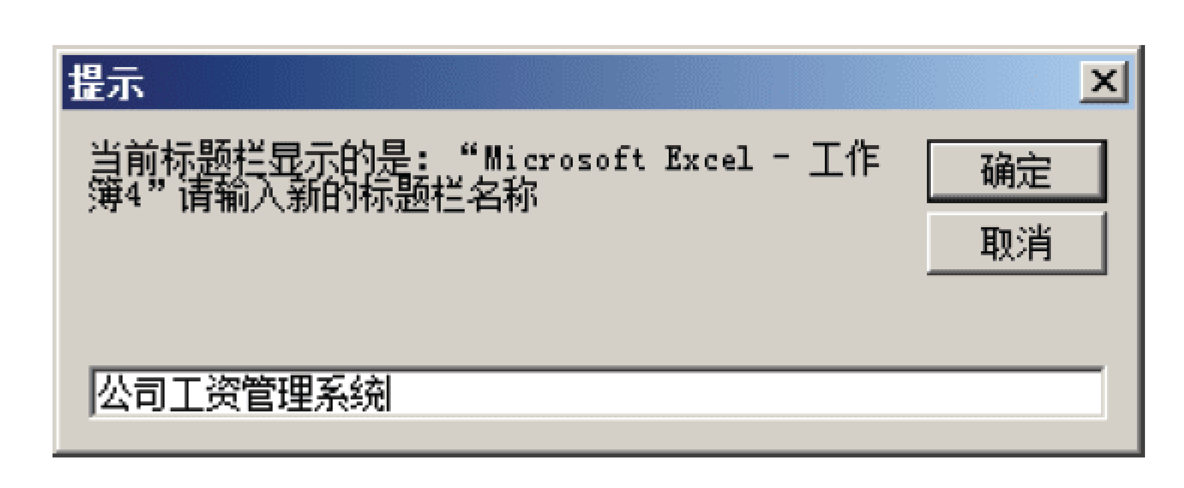


图 10.11　输入对话框

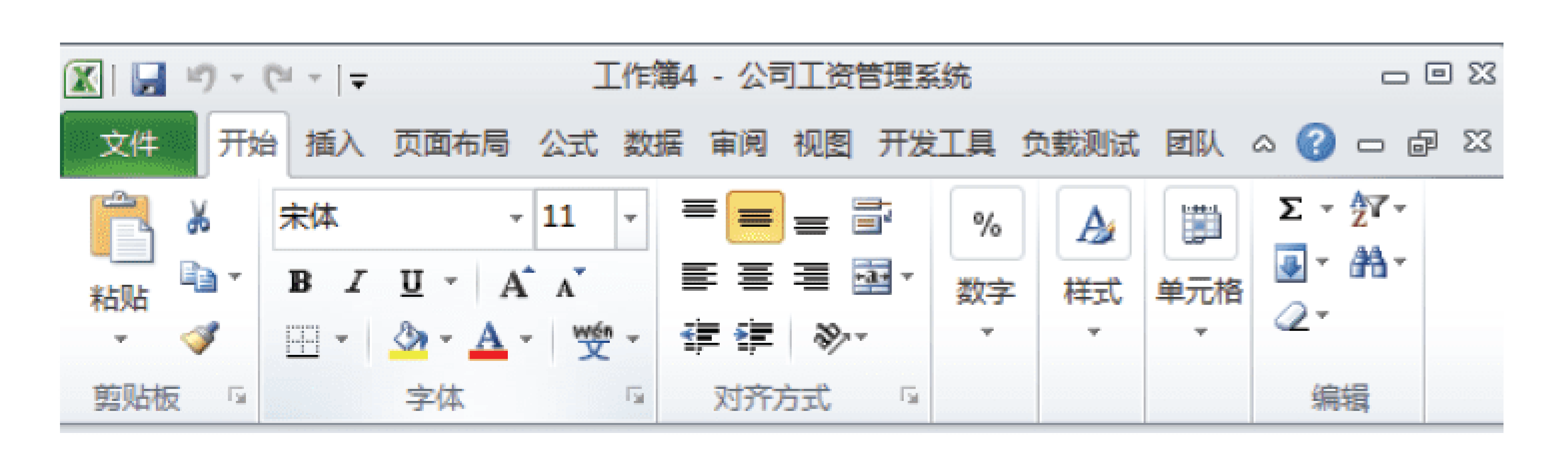


图 10.12　标题栏被更改

【代码解析】本段代码首先读取 Application 对象的 Caption 属性获得当前标题栏名称。使用 InputBox 函数来获取用户输入文字后将其赋予 Caption，从而实现了对标题栏的修改。

注意：Caption 属性是对 Excel 应用程序本身进行设置，如果关闭了包含上述代码的工作簿，标题栏的更改不会改变。

10.2.2　使用 Excel 界面状态栏

Excel 的状态栏用于显示操作状态，使用 Application 对象的 StatusBar 属性能够返回或设置 Excel 窗口状态栏的文字。同时，使用 DisplayStatusBar 属性可以控制状态栏是否显示。

【范例 10-9】 显示状态栏，并修改状态栏提示信息，代码如下所示。

```
Sub 使用状态栏()
    Dim old As String                                    '声明变量
    old = Application.StatusBar                          '保存状态栏初始状态
    If Application.DisplayStatusBar Then                 '判断是否显示状态栏
        Application.StatusBar = "数据正在处理中，请稍候…… "
                                                     '显示则更改状态栏文字
    Else
        Application.DisplayStatusBar = True          '不显示，则使状态栏显示
        Application.StatusBar = "数据正在处理中，请稍候……"
                                                     '显示指定文字
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，状态栏文字被修改，如图 10.13 所示。

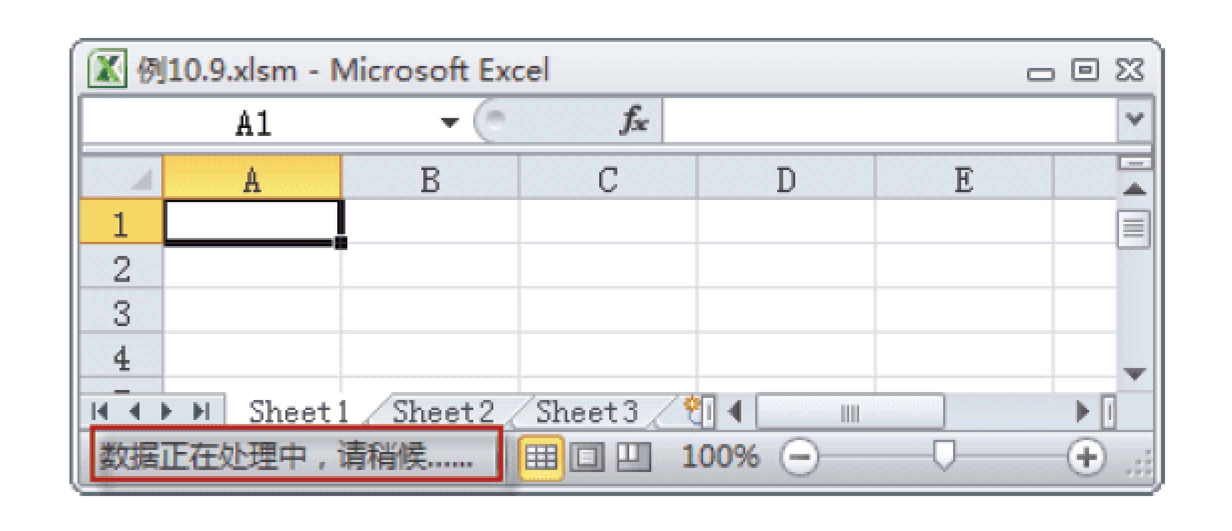


图 10.13　修改状态栏信息

【代码解析】本段代码演示获取状态栏状态和更改状态栏显示文字和状态的方法。在 Excel VBA 中，Application 对象的 DisplayStatusBar 属性决定了状态栏是否显示，当其为 True 时显示状态栏，当其为 False 时状态栏不显示。在第 04 行中使用 IF 语句判断状态栏是否显示，如果显示，则直接通过修改 StatusBar 属性值来更改状态栏显示的文字。否则，将 DisplayStatusBar 属性值设置为 True 使状态栏显示，然后再修改其显示文字。

注意：在程序中如果需要将修改后的状态栏恢复到初始状态，则可在修改前将状态栏状态保存在一个变量中，如本例中第 03 行代码。保存初始状态后，当需要恢复到初始状态，使用下面语句即可：

```
Application.StatusBar=old。
```

10.2.3　设置 Excel 窗口最大化和最小化

在 Excel 窗口中，单击左上角的“最小化”按钮可将程序窗口最小化，而单击“最大化”按钮可使程序窗口最大化。在使用 VBA 编程时，通过读取 Application 对象的 WindowState 属性的属性值，可以查询 Excel 窗口的状态，同时改变该属性值能够改变窗口的状态。

【范例 10-10】 编程实现 Excel 应用程序窗口状态的改变，代码如下所示。

```
Sub 修改窗口状态()
    If Application.WindowState = xlMaximized Then  判断窗口是否最大化
        Application.WindowState = xlNormal          '窗口设置为正常窗口
    ElseIf Application.WindowState = xlNormal Then '判断窗口是否为
                                                    普通窗口
        Application.WindowState = xlMinimized       '如果是，则窗口最小化
    Else
        Application.WindowState = xlMaximized
                                            '窗口非最大化状态，窗口最大化
    End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，Excel 窗口将根据当前状态改变窗口大小。如当前窗口为最大化状态，窗口将变为正常状态，如图 10.14 所示。

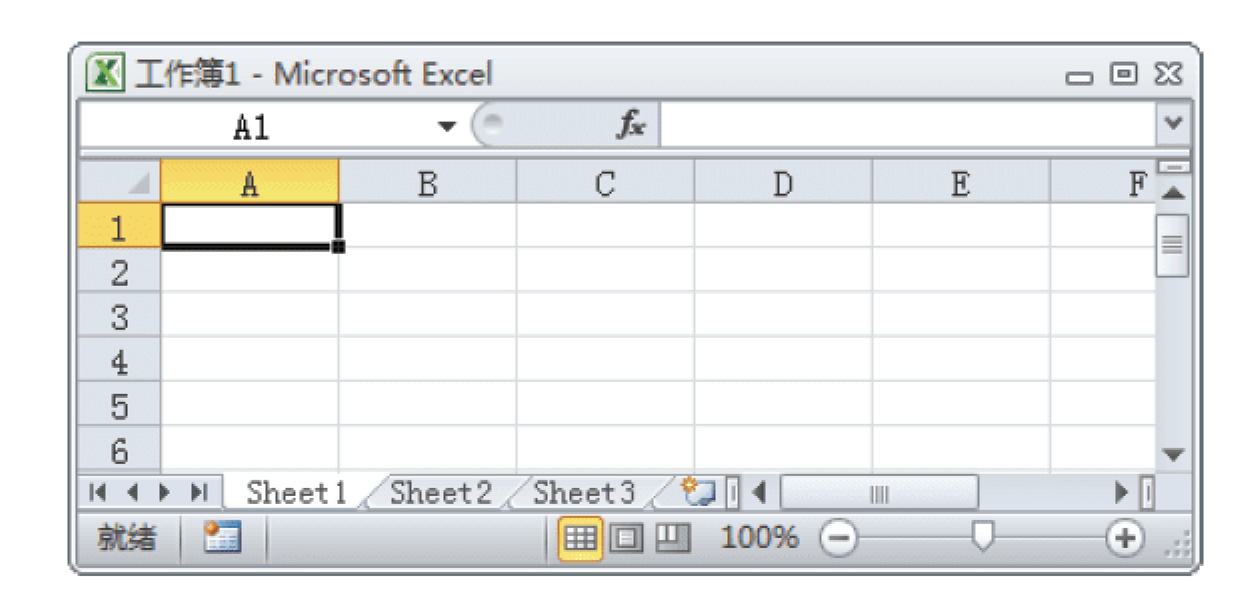


图 10.14　窗口正常状态

【代码解析】本示例程序用于演示编程改变 Excel 应用程序窗口状态的方法，Application 对象的 WindowState 属性可以设置为 3 个常量值，它们分别是 xlMaximized（窗口最大化）、xlMinimized（窗口最小化）和 xlNormal（正常窗口）。在本实例中使用 If 结构来对 WindowState 属性值进行判断，根据属性值的不同来更改程序窗口的状态。

10.2.4　设置 Excel 界面光标形状

Excel 默认的光标是，这个光标可以通过编写 VBA 程序来更改。使用 Application 对象的 Cursor 属性，能够设置光标在 Excel 窗口中的形状，可以将光标指针更改为工形、箭头形以及沙漏形。

【范例 10-11】 编写 VBA 程序更改 Excel 光标针，代码如下所示。

```
Sub 设置鼠标形状()
    Application.Cursor = xlNorthwestArrow          '光标设置为箭头形
    MsgBox "您现在使用的是箭头形光标，_
    光标放于工作表中可以查看其形状。"              '给出指针形状提示
    Application.Cursor = xlIBeam                   '光标设置为工形
    MsgBox"您将使用工形指针光标放于工作表中可以查看其形状。"
    Application.Cursor = xlWait                    '光标设置为沙漏形
    MsgBox "您将使用沙漏形光标,光标放于工作表中可以查看其形状。"
    Application.Cursor = xlDefault                 '光标设置为默认形状
    MsgBox "您的光标已经恢复为默认状态，光标放于工作表中可以查看其形状。"
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，Excel 界面中光标被设置为指定的形状，同时将提示当前光标的状态，如图 10.15 所示。

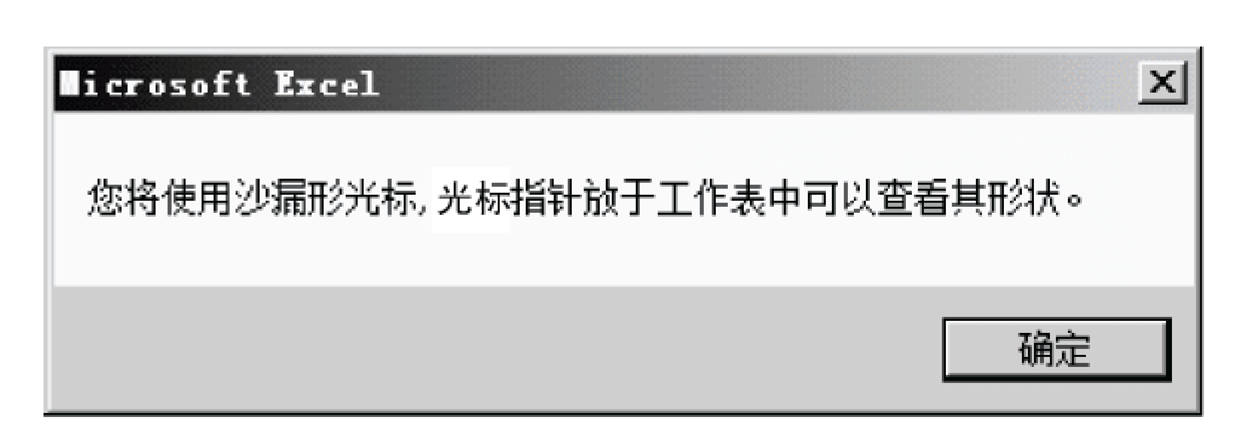


图 10.15　光标形状改变并提示

【代码解析】本段程序通过设置 Cursor 属性来更改光标的形状，Cursor 属性值为 4 个常量，本段代码都用到了。通过本段代码，读者能了解各个常量对应的光标形状。

注意：程序运行停止后，光标不会自动恢复到默认的状态。如果需要默认指针，应该在退出程序时将光标样式改回来。

10.3　操作 Excel 单元格

单元格是 Excel 工作表的基本元素，数据的操作离不开单元格的操作。使用 Application 的属性和方法能够很方便地实现对 Excel 工作表中的单元格的各项操作，本节将介绍几种常见操作的实现方法。

10.3.1　快速选择 Excel 单元格

使用 Application 对象的 Goto 方法能够选定任意工作簿中的任意单元格区域。此时，如果该工作簿未处于活动状态，该工作簿将会被激活。Goto 方法的语法格式如下：

```
表达式.Goto Reference=参数,Scroll：=参数
```

参数说明如下所示。

- 表达式：一个代表 Application 对象的变量。
- Reference：指明目标区域。其“=”后的参数可以是 Range 对象、包含 R1C1 样式的单元格引用字符串或包含 Visual Basic 过程名的字符串。如果省略该参数，目标将为最近一次用 Goto 方法选定的区域。
- Scroll：其“=”后的参数可以是 True 或 False。如果为 True，则滚动窗口直至区域的左上角出现在窗口的左上角中。如果为 False，则不滚动窗口。默认值为 False。

【范例 10-12】 快速选择“sheet3”工作表中的“A1:B5”区域，代码如下所示。

```
01  Sub Goto 方法的使用()
02      Aplication.Goto Reference:=Worksheets("sheet3").Range("A1:B5"), _
03      scroll:=True                                   '跳到工作表的指定单元格区域
04   End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，“Sheet3”工作表中指定的单元格被选择，如图 10.16 所示。

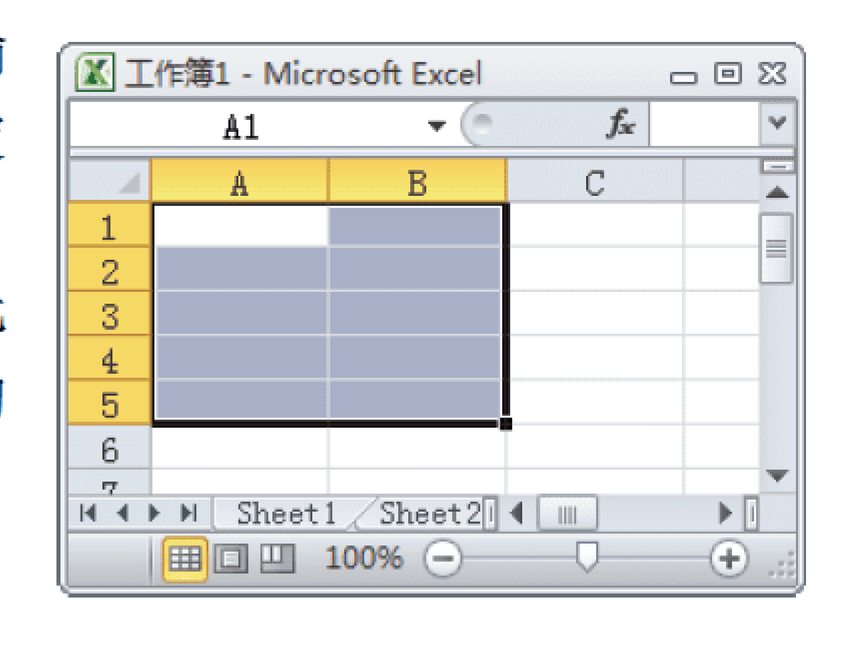


图 10.16　指定单元格被选择

【代码解析】本段程序演示使用 Goto 方法来选择单元格区域的方法。程序代码的第 02 行通过设置 Goto 方法的 Reference 参数和 Scroll 参数实现向 sheet3 工作表的 A1：B5 区域的跳转并滚动窗口至区域的左上角。

注意：VBA 中使用 Select 方法也能实现单元格的选选取。使用 Goto 方法时，如果指定的区域不在最前面屏幕的工作表中，Microsoft Excel 将在选定该区域之前切换至该工作表。而对不在屏幕的最前面的工作表中的区域，使用 Select 方法则只会选定该区域并不激活该工作表。Goto 方法有让用户滚

动目标窗口的 Scroll 参数，而 Select 方法没有。在实际的编程过程中，使用哪种方法，用户可根据需要选择。

10.3.2　同时选择 Excel 多个区域单元格

在创建数据表时，往往需要同时向多个单元格中输入数据。这里使用 Union 方法，可以方便地在程序中完成这个任务。Union 能够返回两个或多个单元格区域的合并区域，其语法格式如下：

```
表达式.Union(Arg1, Arg2,…… )
```

参数说明如下所示。

- ❑ 表达式：一个代表 Application 对象的变量。
- ❑ Arg1，Arg2…… ：指定单元格区域的参数。

【范例 10-13】 在工作表中向负数单元格填充颜色，代码如下所示。

```
01  Sub 合并单元格()
02      Dim rngA As Range, rngB As Range              '声明对象变量
03      For Each rngA In Range("A2:D20")              '遍历指定单元格
04          If rngA < 0 Then                          '如果单元格为负数
05              If rngB Is Nothing Then               '如果没有指定对象
06                  Set rngB = rngA                   '对象变量赋值
07              Else
08                  Set rngB = Application.Union(rngB, rngA)   '组合单元格
09              End If
10          End If
11      Next
12      rngB.Interior.ColorIndex = 15                 '向组合单元格填充颜色
13  End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，sheet1 工作表中指定的单元格将填入数字，如图 10.17 所示。

【代码解析】 本段程序使用 Application 对象的 Union 方法将多个单元格合并为一个单元格区域，并同时向该区域填充颜色。程序使用 For…Nxet 循环遍历 A2 至 D20 单元格区域的所有单元格，判断单元格是否为负数，如果为负数，则使用 Union 方法将找到的单元格合并为 1 个单元格。这里，在第一次循环时，对象变量 rngB 为空，第 04 行代码判断其是否为空，如果为空，则将 rngA 对象变量的值赋予它。每一次找到符合条件的单元格后，第 08 行代码都将该单元格与原来的合并单元格 rngB 合并一次后赋予 rngB。

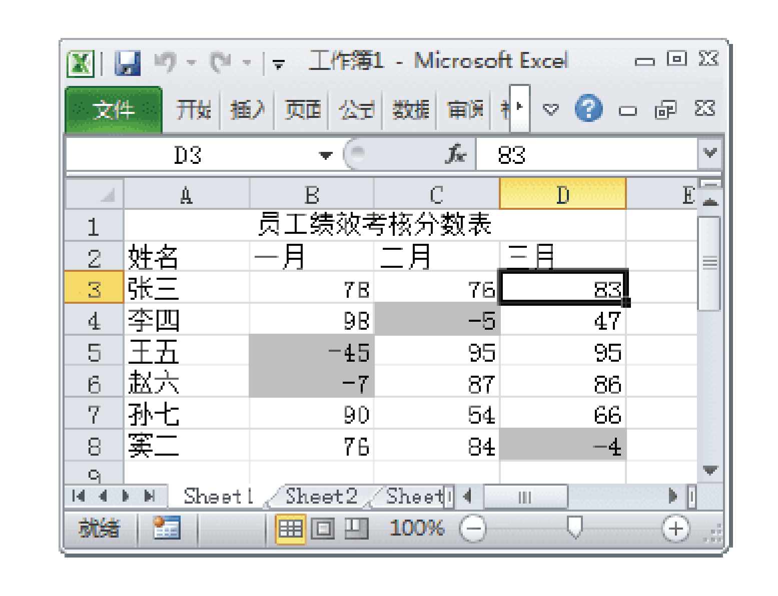


图 10.17　指定单元格填充了数字

警告：在使用 Union 方法时，指定的单元格区域不能少于 2 个，否则就会出错。

10.3.3　取消复制或剪切操作

数据的复制和剪切是创建数据表的常见操作。在 VBA 中，对单元格进行了复制或剪切操作后，使用 CutCopyMode 属性可以获得操作状态，并通过更改该属性的值来实现取消对单元格的复制或剪切操作。

【范例 10-14】编写程序取消对选择单元格数据的复制或剪切操作，代码如下所示。

```
Sub 取消复制或剪切操作()
    Select Case Application.CutCopyMode          '判断复制剪切状态
    Case Is = False                              如果没有复制剪切操作
        MsgBox "没有进行复制或剪切操作！"
    Case Is = xlCopy
        MsgBox "您已经进行了复制操作！"           '如果进行了复制操作
    Case Is = xlCut                              '如果进行了剪切操作
        MsgBox "您已经进行剪切操作！"
   End Select
   Application.CutCopyMode = False               '取消复制剪切操作
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。选择单元格，按 Ctrl+C 复制数据，切换到 VBA 编辑中按 F5 键运行程序。此时程序给出提示对话框，如图 10.18 所示。关闭对话框后，程序将取消当前的复制状态，无法再实现粘贴操作。如果对选择单元格进行了剪切操作，程序运行结果与复制时类似。而如果未进行剪切复制操作，程序给出提示，如图 10.19 所示。

【代码解析】本段程序演示使用 CutCopyMode 属性来控制复制和剪切操作的方法。Application 对象的 CutCopyMode 属性值反映当前复制和剪切操作的状态，其值有 3 个，分别是 False、xlCopy 和 xlCut，这 3 个属性值具体的含义可以通过本实例的运行进行了解。本实例利用 Select...Case 结构来对 CutCopyMode 属性进行判断，根据不同的状态显示不同的对话框内容。当将 CutCopy 属性设置为 False 或 True 时，将取消复制和剪切状态，同时将清除移动边框。

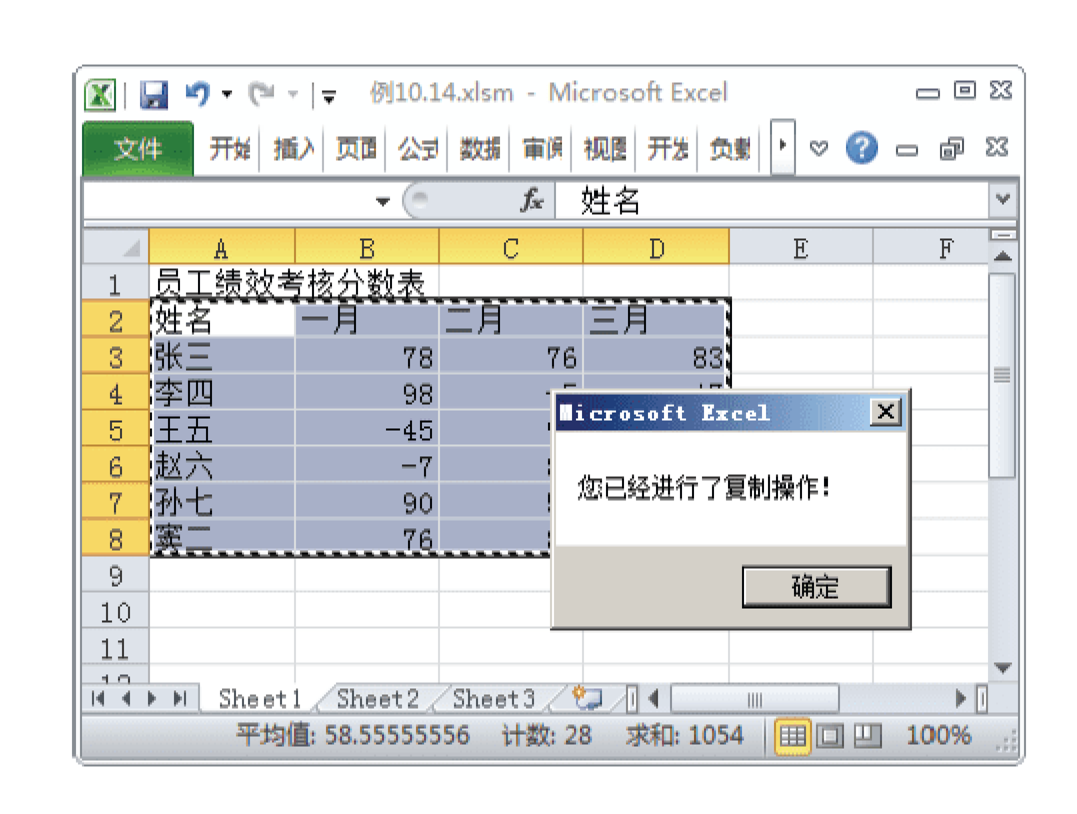


图 10.18　程序给出提示

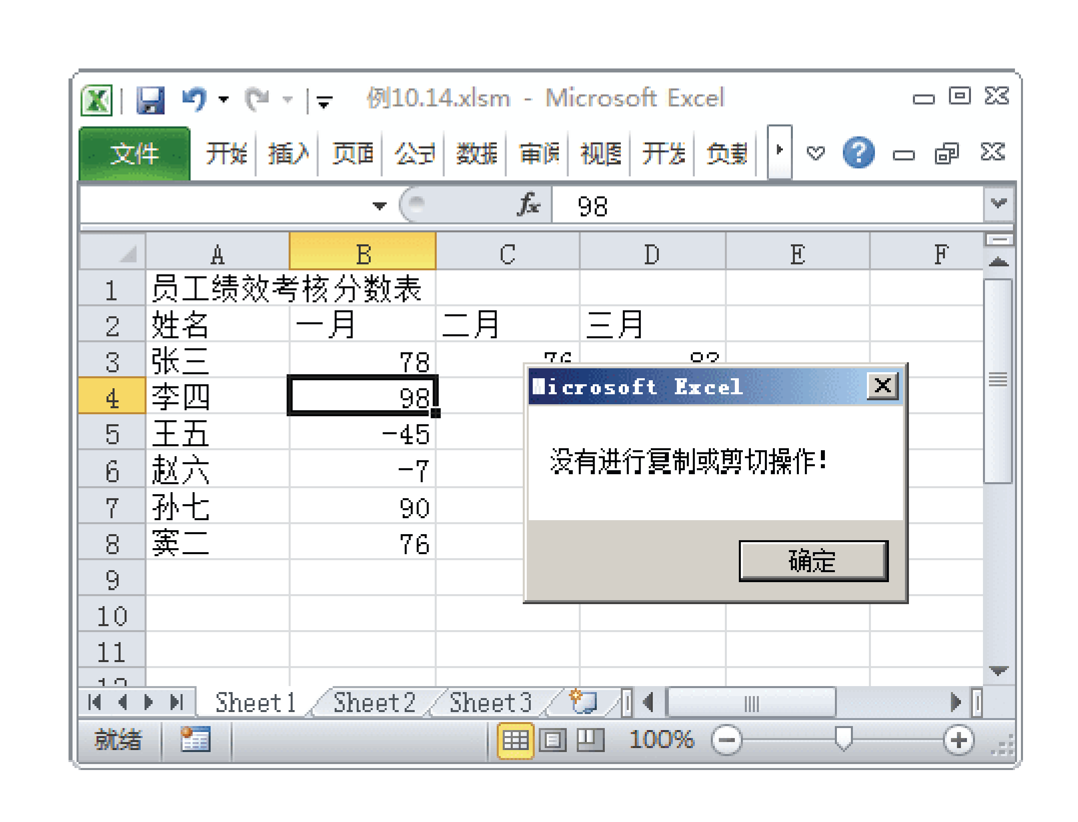


图 10.19　取消复制状态

10.4　操作 Excel 文件

使用 Excel，离不开文件的操作，这包括获取打开或保存文件的文件名以及对文件进行保存等操作。使用 Application 对象的 GetOpenFilename 方法、GetSaveAsFilename 方法以及 FileDialog 对象的方法能够方便地实现对文件的上述操作。

10.4.1　获取文件夹中指定文件的 Excel 文件名

Application 对象的 GetOpenFilename 方法能够获得一个文件的文件名。在使用该方法时将显示一个“打开”对话框，但使用对话框并非打开文件而只是获取选择文件的文件名。GetOpenFilename 方法的语法格式如下：

```
表达式.GetOpenFilename(FileFilter, FilterIndex, Title, ButtonText, MultiSelect)
```

参数说明如下所示。

- 表达式：一个代表 Application 对象的变量。
- FileFilter：该参数是一个指定文件筛选条件的字符串。该参数是 Variant 数据类型，是可选参数。
- FilterIndex：该参数用于指定默认文件筛选条件的索引号，取值范围为 1 到由 FileFilter 所指定的筛选条件数目。如果省略该参数，或者该参数的值大于可用筛选条件数，则使用第一个文件筛选条件。
- Title：指定对话框的标题。如果省略该参数，则标题为“打开”。
- ButtonText：此参数仅限于 Macintosh 机（即苹果个人计算机中的一个系列）使用。
- MultiSelect：当此参数为 True 时，则允许选择多个文件名。如果为 False，则只允许选择一个文件名。默认值为 False。

【范例 10-15】获取选择文件夹中指定文件类型的文件名列表，代码如下所示。

```
Sub 获取文件名()
    Dim n As Integer, filename As Variant, a As Variant     '声明变量
    filename = Application.GetOpenFilename(fileFilter:= _
    "Word 文档 (*.doc),*.doc, Excel 启用宏文件的工作簿(*.xlsm),* .xlsm" _
    , MultiSelect:=True)                                    '获取文件名
    For Each a In filename                                  '遍历所有文件名
        n = n + 1                                           '指针加 1
        Cells(n, 1) = a                                     '文件名填入单元格
    Next
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，此时程序打开“打开”对话框，在“文件类型”下拉列表框中显示程序中设置的两种文件类型。按 Shift 键同时选择需要的文件，如图 10.20 所示。单击“确定”按钮后，选择文件的保存路径及其文件名依次填入单元格中，如图 10.21 所示。

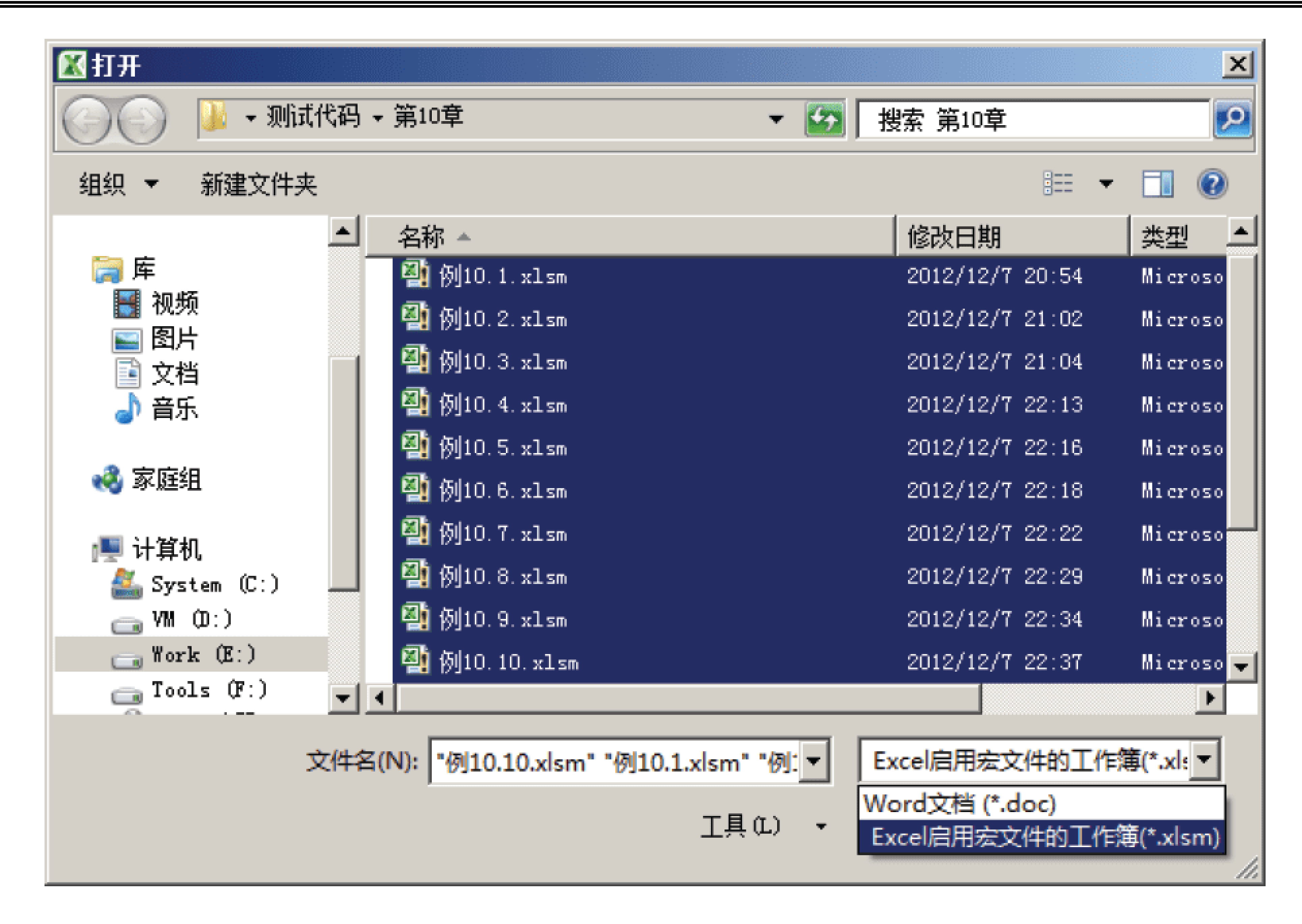


图 10.20 “打开”对话框

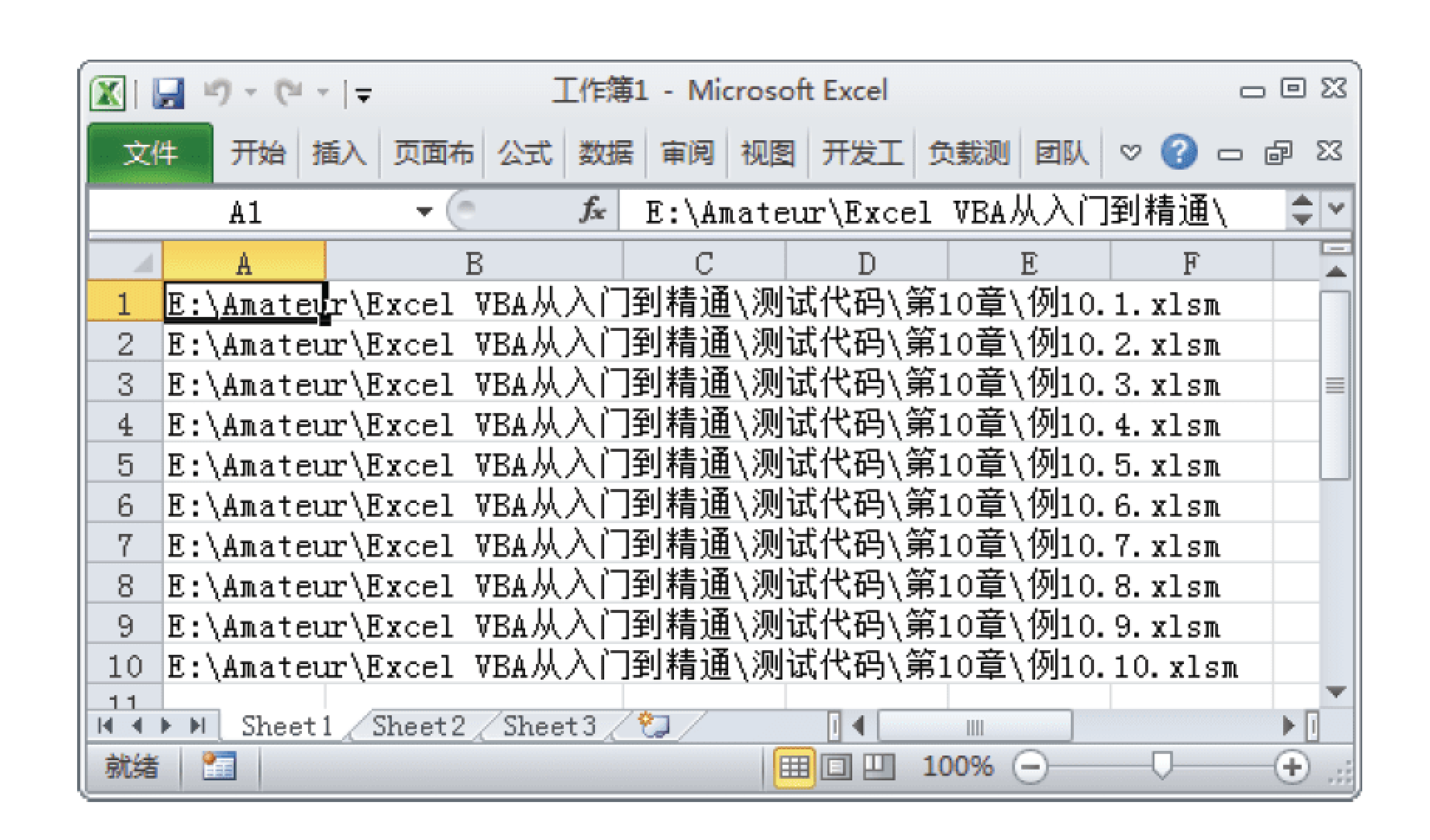


图 10.21 单元格中显示文档保存路径后的文件名

【代码解析】本范例使用 GetOpenFilename 方法获取在“打开”对话框中选择的文件的文件名，使用 For Each…In Next 结构遍历获得的文件名，并将文件名置于指定的单元格中。在使用 GetOpenFilename 方法时，设置 fileFilter 参数指定文件类型，设置 MultiSelect 参数允许同时选择多个文件。这里要注意，没有选择文件就关闭“打开”对话框时，程序会出错，此时可以像示例那样在第 06 行增加一个错误处理程序。

提示： 在设置 fileFilter 参数时，可以指定多个文件类型。FileFilter 参数中传递的字符串由文件筛选字符串对以及后面的 MS-DOS 通配符文件筛选规范组成，中间以逗号分隔。如果该参数省略，则该参数默认为“所有文件 (*.*),*.*”。

10.4.2　获取 Excel 文件保存位置

使用 Application 对象，能够打开标准的“另存为”对话框获得文件保存的位置和保存文件名。GetSaveAsFilename 方法的语法结构如下：

```
表达式.GetSaveAsFilename(InitialFilename, FileFilter, FilterIndex, Title,
ButtonText)
```

其参数意义与 GetOpenFilename 方法基本相同，只是多了一个 InitialFilename 参数，该参数用于指定建议的文件名。如果省略该参数，Microsoft Excel 使用活动工作簿的名称。

【范例 10-16】获取选择文件中指定文件类型的文件名列表，代码如下所示。

```
Sub 获取保存文件位置()
    Dim fileSaveName As Variant                    '变量声明
    fileSaveName = Application.GetSaveAsFilename( _
    fileFilter:="Excel 启用宏的工作簿(*.xlsm), *.xlsm")
                                                   '获取文件保存路径和文件名
    If fileSaveName <> False Then                  '如果获取文件名成功
        MsgBox "文件保存为： " & fileSaveName      '显示提示信息
    Else
        MsgBox "你已经取消了保存，无法获得文件名"  '没有获得文件名，给出提示
   End If
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，此时程序打开“另存为”对话框，选择文件保存的位置，并在“文件名”文本框中输入保存文件的文件名，如图 10.22 所示。

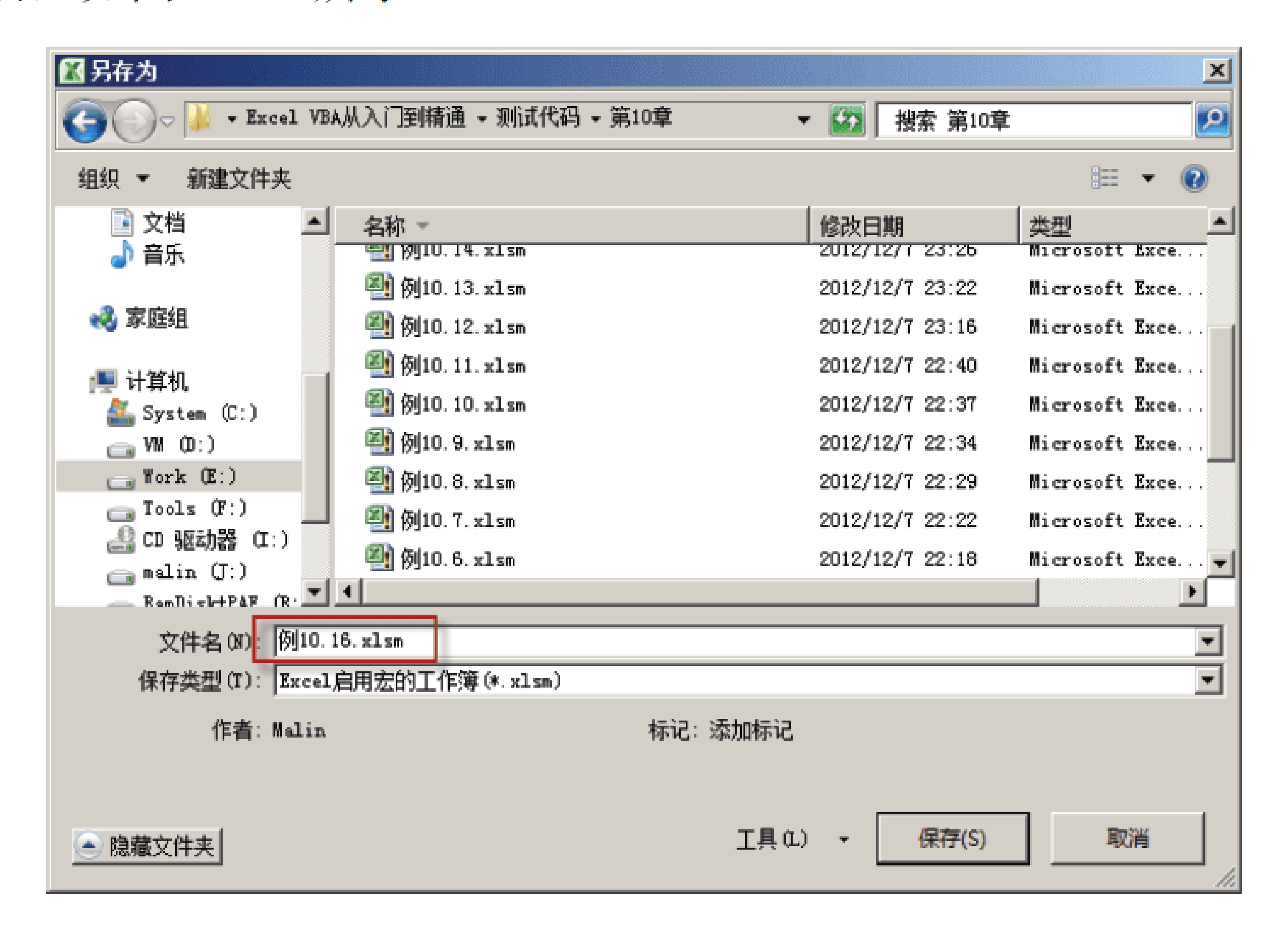


图 10.22　打开的“另存为”对话框

单击“确定”按钮关闭对话框后，程序提示文件保存路径和文件名，如图 10.23 所示。如果单击了“取消”按钮，程序同样给出提示，如图 10.24 所示。




图 10.23　提示文件保存路径和文件名



图 10.24　提示取消了保存

【代码解析】在范例中，使用 GetSaveAsFilename 方法的打开“另存为”对话框获取文件保存路径和文件名。由于 GetSaveAsFilename 方法返回值为 Variant 数据类型，在代码的第 02 行将变量声明为该类变量。如果用户单击了 GetSaveAsFilename 方法打开的“另存为”对话框的“确定”按钮，该方法将返回详细的文件保存路径和文件名，否则将返回 False。因此，在代码中使用 If 语句来对用户的操作进行判断，并根据判断给出不同的提示。

🔔警告：使用本方法获取文件保存路径和文件名时，由于需要选择磁盘和文件夹，将会更改 Excel 默认的文件保存位置，这是读者在操作时要注意的。

10.4.3　打开最近使用的 Excel 文件

使用 Application 对象所属的 RecentFiles 对象集合的 Name 属性能够获得最近使用文件的文件名列表，而使用 Open 方法能够根据文件名打开指定的文件。下面通过一个示例来介绍打开最近使用的文件的方法。

【范例 10-17】 显示最近使用过的 3 个文件的信息，并打开最近使用的第二个文件，代码如下所示。

```
Sub OpenRecentFiles()
    MsgBox "最近使用的第一个文件为：:" & Application.RecentFiles(1).Name
    & Chr(13) & "最近使用的第二个文件为：" & Application.RecentFiles(2).Name _
    & Chr(13) & "最近使用的第三个文件为："& Application.RecentFiles(3).Name _
    & Chr(13) & "单击“确定”按钮后打开第二个文件。", _
    vbOKOnly, "显示最近使用的 3 个文件"                '显示最近使用的文件名称
    Application.RecentFiles(2).Open                  '打开最近使用的第一个文件
End Sub
```

【运行结果】插入一个模块，在模块的“代码”窗口输入以上代码。按 F5 键运行程序，程序将列出最近打开的 3 个文件，如图 10.25 所示。关闭对话框后列表中的第二个文件将被打开。

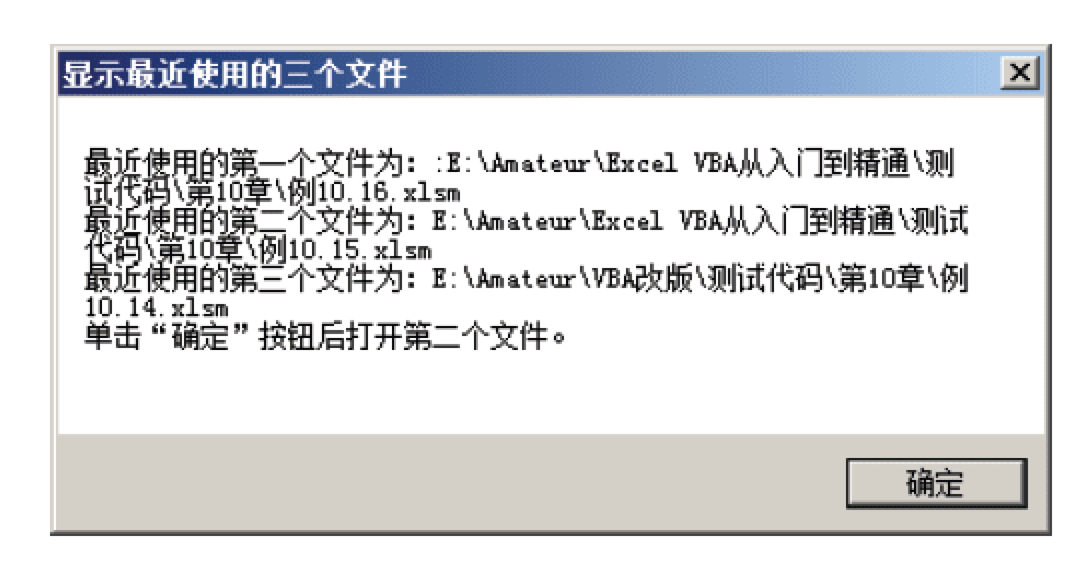


图 10.25　显示最近打开的文件

【代码解析】在范例中，使用 MsgBox 显示最近打开的 3 个文件的文件名。RecentFiles（文件编号）语句将返回由文件编号指定的 RecentFile 对象，即曾经打开过的文件。在第 07 行，使用 Open 方法打开由 RecentFiles(2)语句指定的文件。

警告：在使用 Open 方法打开文档时，如果文档处于打开状态，程序将会报错。

10.5　小　　结

Application 是 Excel VBA 的顶层对象，其包括了众多的对象成员，本章着重介绍了 Application 对象的方法和属性在编程中的使用技巧。通过本章的学习，读者将能够掌握通过编程实现 Excel 常见操作的方法，了解 Excel 程序窗口的设置方法，掌握常见的单元格操作和文件操作的技巧。

Application 对象成员很多，本章不可能逐一介绍，读者可以查阅 VBA 帮助来获得相关的资料。同时，由于在实际的程序设计中，大多数的事件响应程序都使用 Workbook 和 WorkSheet 对象的事件响应来编写事件代码，而且在编程中直接使用它们的事件将更为方便快捷，因此本章没有专门介绍 Application 对象的事件。

10.6　本 章 习 题

1．下面哪个方法能够打开“打开”对话框？（　　）

A．FindFile 方法　　B．GoTot 方法　　C．InputBox 方法　　D．Open 方法

2．选择下面语句的运行的结果。（　　）

```
Application.OnTime
EarliestTime:=TimeValue("18:00:00"),Procedure:="my_Procedure", _
 Schedule:=False
```

A．在 18 点启动 my_Procedure 过程

B．取消在 18 点启动 my_Procedure 过程

C．程序运行 18 秒后启动 my_Procedure 过程

D．程序运行 18 秒后启动 my_Procedure 过程，不考虑 Schedule 中的对时间的设定

3．运行下列过程后，Excel 应用程序窗口会有怎样的变化？（　　）

```
Sub test()
    Dim myString As String, a As String
    a = Application.Caption
    myString = InputBox("当前标题栏显示的是：" & "“" & a & "”", _
    " & "请输入新的标题栏名称", "提示")
     Application.Caption = myString
     Application.StatusBar = Application.UserName
     Application.WindowState = xlMaximized
End Sub
```

A．标题栏标题变为输入文字，状态栏显示程序名，窗口最大化

B．标题栏标题变为文档名，状态栏显示程序名，窗口最小化

C．标题栏标题变为输入文字，状态栏显示系统用户名，窗口最大化

D．标题栏标题变为文档名，状态栏显示文档名，窗口最大化

4．对下面程序描述正确的是：（　　）

```
Sub test()
    Dim n As Integer, filename As Variant, a As Variant
    "Word 文档 (*.doc),*.doc, Excel 启用宏文件的工作簿(*.xlsm), _
    *.xlsm, MultiSelect:=True)
    For Each a In filename
        n = n + 1
        Cells(1, n) = a
    Next
End Sub
```

A．在“打开”文本框中只能显示 Excel 启用宏文件的工作簿文档

B．在“打开”对话框中选择的第一个文件的文件名将写入工作表的 A1 单元格中

C．程序将能够打开在“打开”对话框中选择的文件

D．在“打开”对话框中将能够显示 Word 文档或 Excel 启用宏文件的工作簿文档

5．编写程序显示 Excel 的内置对话框。

【提示】Application 对象成员中有一个 Dialogs 属性，该属性能够返回 Dialogs 对象集合，能够获得 Excel 的内置对话框，使用该对话框能够实现相关的操作。

6．编写程序逐条显示在“打开文件”对话框中选择的文件。

【提示】Application 对象的 FileDialog 属性能够返回一个 FileDialog 对象，该对象可以实现对文件对话框的控制。使用 FileDialog 对象的 SelectedItems 属性获得 FileDialogSelectedItems 集合，使用集合对象的 Count 方法来获得选择文件的个数。使用 For…Next 循环结构来实现逐条显示文件信息，循环次数的上限是获得的选择文件的个数。

第 11 章　工作簿对象

Workbook 是 Application 对象下的子对象，它对应的是 Excel 工作簿。工作簿其实就是一个 Excel 文件，通过 Workbook 对象可以操作工作簿，主要包括引用工作簿、激活工作簿、保存工作簿、为工作簿设置密码、通过工作簿的事件响应机制，可以响应对工作簿的相关操作，例如激活工作簿引发的事件、关闭工作簿引发的事件。本章的主要学习内容和学习目标如下：

- 掌握使用工作簿对象新建和打开 Excel 工作簿的方法；
- 掌握使用工作簿对象保存 Excel 工作簿的方法；
- 掌握使用工作簿对象保护 Excel 文档的方法；
- 掌握通过工作簿事件响应对 Excel 工作簿的操作方法。

11.1　引用 Excel 工作簿对象

在 Excel 中，每一个文档实际上就是一个工作簿对象。工作簿对象可以包含多个工作表，是容纳工作表的容器。在 VBA 中，Workbook 对象是 Workbooks 集合的成员，Workbooks 对象集合包含了所有在 Excel 中打开的 Workbook 对象。

11.1.1　引用 Excel 工作簿的方法

要对工作簿进行操作，首先要引用工作簿。VBA 程序中引用工作簿可以通过使用索引号和工作簿名来实现。使用索引号引用工作簿的语法格式如下：

```
Workbooks（索引号）
```

这里，索引号是工作簿的编号，是按照创建或打开的顺序排列的。如，在打开多个工作簿时，如果需要引用第 2 个工作簿可以使用下面的语句：

```
N=Workbooks（2）.Name
```

如果需要引用最后一个工作簿，可以使用 Count 属性来获得工作簿的数量，这样就可以使用下面的语句来引用最后一个工作簿。

```
N=Workbooks(Workbooks.Count).Name
```

使用索引号来引用工作簿的方法并不方便，因为对于普通用户来说很难准确获知工作簿的索引号。实际上使用工作簿名称能够方便地实现工作簿的引用。工作簿的名称可以通过 Workbook 的 Name 属性来获得，下面通过一个实例来介绍使用工作簿名来引用工作簿

的方法。

【范例 11-1】 使用提示对话框显示打开的工作簿名称，代码如下所示。

```
01  Sub 显示工作簿名()
02      a = Workbooks.Count                '获得打开工作簿个数
03      b = Workbooks(1).Name              '获得第一个工作簿名
04      c = Workbooks(a).Name              '获得最后一个工作簿名
05      MsgBox "当前打开的工作簿一共有" & a & "个" & Chr(13) & _
06      "第一个是：" & b & Chr(13) & _
07      "最后一个是：" & c                  '在提示对话框中显示工作簿个数和名称
08  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序给出提示信息，如图 11.1 所示。

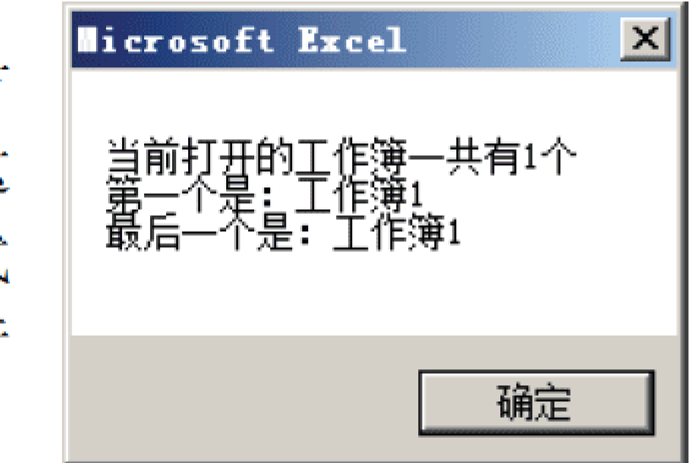


图 11.1　显示工作簿名

【代码解析】本段代码演示在 VBA 程序中引用工作簿的方法。代码的第 02 行使用 Count 方法获得打开工作簿的个数，这里一共打开了 1 个工作簿。第 03 行和第 04 行使用索引号方式引用工作簿对象，获得打开的工作簿名称。第 05 行至第 07 行使用 MsgBox 函数来显示打开的工作簿个数和名称。

提示： 从程序可以看到，工作簿名实际上就是文档名。使用工作簿名引用工作簿对象时，工作簿名需要放在引号中。另外工作簿名也可以是变量或者表达式。

11.1.2　激活 Excel 工作簿对象

当打开了多个工作簿时，如果需要在程序中对某个工作簿进行操作，必须先激活这个工作簿，激活一个工作簿的目的是将该工作簿作为当前处理的工作簿。在 Excel VBA 中，使用 Activate 方法来实现对工作簿的激活，语法格式如下所示。

```
Workbooks(index).Activate
```

在 VBA 中，可以使用 ThisWorkbook 和 ActiveWorkbook 来实现对工作簿的引用。这里 ThisWorkbook 是表示当前正在处理的工作簿对象。而 ActiveWorkbook 指的是在打开多个工作簿时被激活的活动工作簿。

【范例 11-2】 编程激活工作簿对象，比较 ThisWorkbook 和 Activebook 所指对象的不同，代码如下所示。

```
Sub 工作簿对象的激活()
    Debug.Print "当前打开的工作簿共有"; Workbooks.Count'显示打开的工作簿数量
    Workbooks(Workbooks.Count).Activate            '激活工作簿
    Debug.Print "当前正在处理的工作簿是：";ThisWorkbook.Name
                                                   '显示正在处理的工作簿名
    Debug.Print "当前激活的工作簿是：";ActiveWorkbook.Name
                                                   '显示激活工作簿名
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，在“立即窗口”显示打开的工作簿名，如图 11.2 所示。

【代码解析】在本示例中，一共打开了 3 个工作簿，编写程序的模块所在的工作簿是“工作簿对象的激活.xlsm”文件。第 03 行激活了这 3 个打开工作簿中的最后一个工作簿。从显示结果可以看到，ThisWorkbook 指向的是正在执行 VBA 过程的工作簿，而 ActiveWorkbook 指向的是使用 Active 方法激活的工作簿。

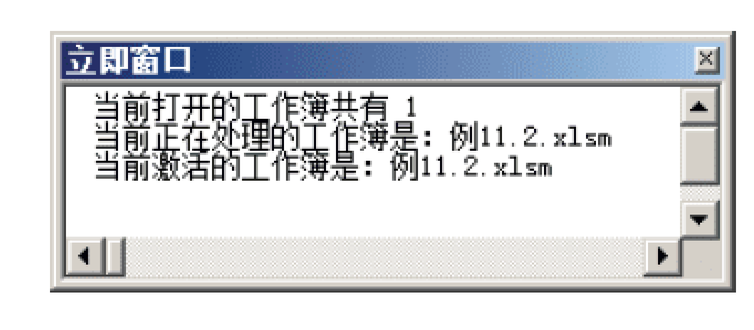


图 11.2 “立即窗口”显示的结果

注意：只有打开的工作簿才能被激活，如果激活一个未打开的工作簿，系统将会给出出错信息。另外，如果子过程只在一个工作簿中执行，并且只处理该子过程所在的工作簿，则 ActiveWorkbook 和 ThisWorkbook 所指的将是同一个工作簿。

11.2　新建和打开 Excel 工作簿

在 VBA 中，Workbooks 对象集包含了在 Excel 应用程序中打开的所有 Workbook 对象。使用 Workbooks 对象集的 Add 和 Open 方法能够在程序中实现工作簿的新建和打开操作。

11.2.1　新建 Excel 工作簿

在 VBA 中创建一个新的工作簿可以使用 Workbooks 对象的 Add 方法来实现。Add 方法的语法结构如下所示。

```
对象.Add(Template)
```

在使用 Add 方法创建新的工作簿时，Template 参数用于指定以哪个文件作为模板来创建工作簿，如果省略该参数将创建一个包含一定数量空白工作表的新工作簿。

【范例 11-3】 创建一个新工作簿并为其命名，代码如下所示。

```
Sub 新建工作簿()
    Dim myBook As Workbook                              '声明对象变量
    Set myBook = Workbooks.Add                          '创建一个新工作簿
    With myBook
        .BuiltinDocumentProperties(1) = "销售部七月业绩表"
                                                        '设置工作簿标题
        .BuiltinDocumentProperties(2) = "销售"          '设置工作簿主题名称
        .SaveAs Filename:="新建的工作簿.xlsx"           '保存工作簿
    End With
End Sub
```

【运行结果】创建一个模块，在该模块的“代码”窗口输入上述代码。按 F5 键运行程序，创建一个名为“新建的工作簿.xlsx”的工作簿，如图 11.3 所示。该工作簿被保存在文档库中，如图 11.4 所示。

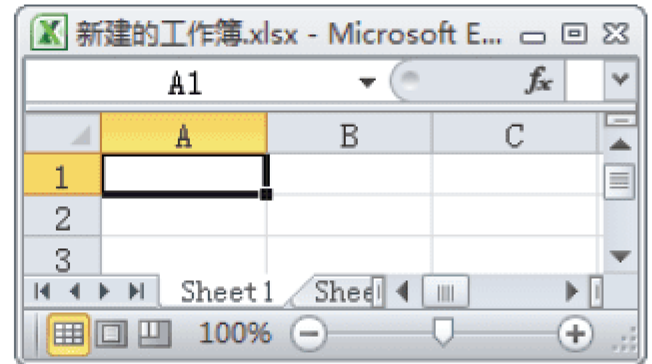


图 11.3　创建新的工作簿



图 11.4　创建文档被保存

【代码解析】本示例演示了使用 Add 方法创建新工作簿的过程。示例中直接使用 Workbooks.Add 语句来创建新工作簿，同时将工作簿对象分配给一个对象变量，使用 With 语句设置新工作簿的属性并保存该文档。

🔔注意：该集合表示指定工作簿的所有内置文档属性。更多信息可查看 VBA 帮助。

11.2.2　打开 Excel 工作簿

在 Excel VBA 中，可以使用 Open 方法来打开工作簿。由于每个工作簿都有保存路径，在打开一个工作簿时必须指明其路径，其语法格式如下：

```
Workbooks.Open "d:\Excel 文档\新工作簿.xls"
```

这里，如果没有指定打开文件的路径，Excel 默认的打开文件路径为“我的文档”。如果没有指定文件名，则程序会打开一个“打开文件”对话框由用户选择需要打开的文件。

如果需要打开带有密码保护的工作簿文件，可以使用下面示例的格式来实现：

```
Workbooks.Open
filename:="D:\book1.xls",password:="abc",writeResPassword:="efg"
```

这里，Password 参数的数据类型是 String，其用于指定文件的打开密码。writeResPassword 参数的数据类型也是 String，用于指定文件的写入密码。

🔔注意：在打开一个带有密码的文件时，如果没有指定相应的参数，Excel 会给出提示对话框要求输入密码。

【范例 11-4】 使用“打开”对话框来打开一个工作簿文件，代码如下所示。

```
Sub 打开文件()
    fl = Application.GetOpenFilename(filefilter : = _
    "Excel 启用宏的工作簿(*.xlsm),*.xlsm")          '获得路径即文件名
    If fl <> False Then Workbooks.Open FileName:=fl _
    ,ReadOnly:=True                                 '打开指定文件
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行

程序，打开“打开”对话框，如图 11.5 所示。在对话框中选择需要打开的文件单击“确定”按钮后，选择的文件被打开。

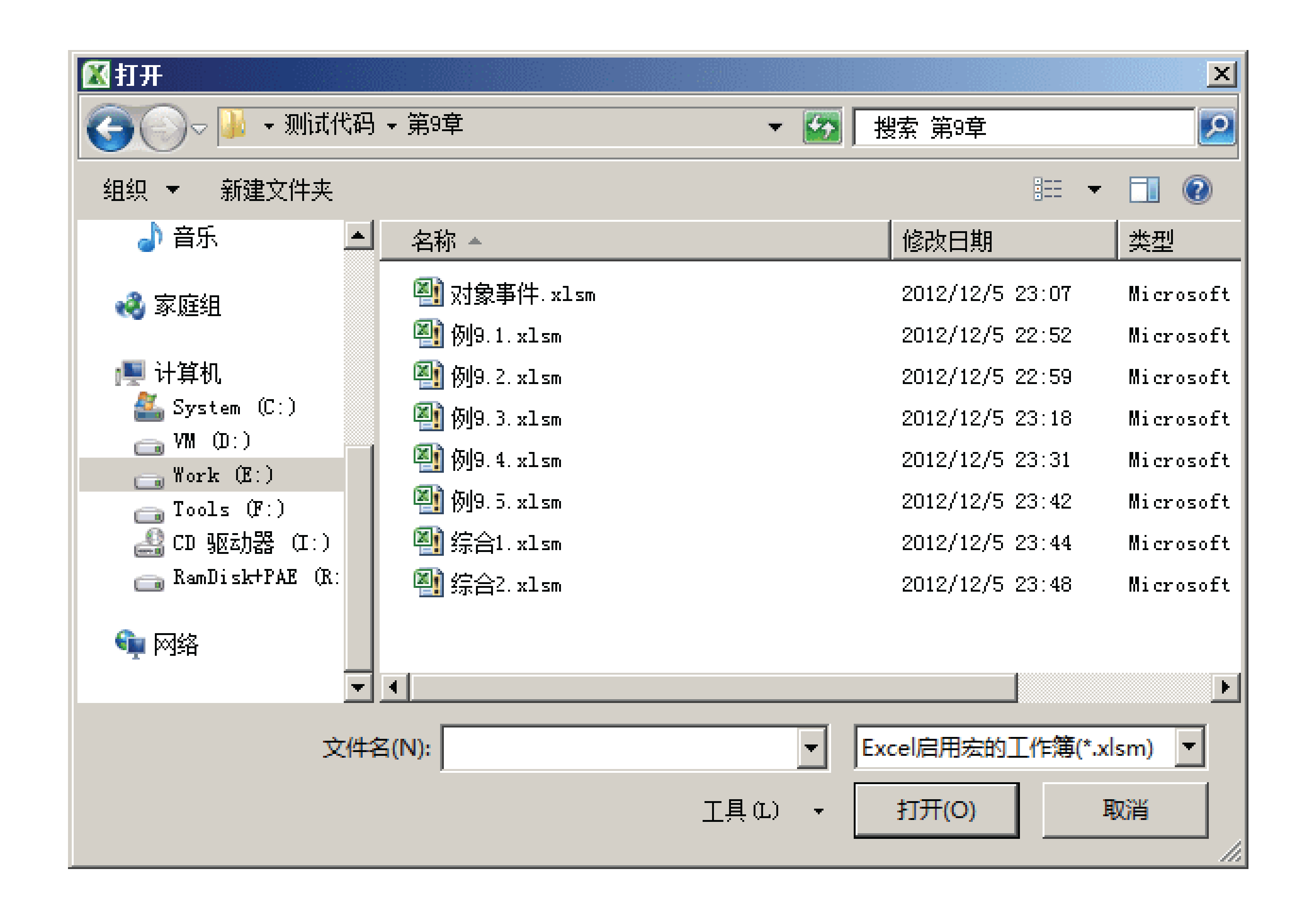


图 11.5　“打开”对话框

【代码解析】本示例演示了编程打开 Excel 文件的方法。示例首先使用 Application 对象的 GetOpenFilename 方法来打开“打开”对话框，用户可以选择需要打开的文件。完成文件选择后，使用 Open 方法打开选择的文件。代码中在使用 Open 方法时，使用了 2 个参数，一个是 FileName 参数，用于指定需要打开的文件。另一个是 ReadOnly 参数，当其值为 True 时，文件将以只读形式打开。

提示：对于打开的工作簿，可以使用 Close 方法来关闭工作簿但不退出工作簿。如果工作簿有改动，Excel 会提示是否保存更改。

11.3　保存 Excel 工作簿

在 Excel 中，保存工作簿有两种方式，一种是保存当前工作簿（即保存已保存的文档），另一种方式是另存为其他工作簿。本节将介绍编程实现这两种保存方式的方法。

11.3.1　使用 Save 方法保存 Excel 工作簿

完成文档操作后往往需要保存文档，在 VBA 中，保存当前工作簿可以使用 Save 方法来实现。该方法能够保存工作簿，但并不会关闭工作簿。如果需要保存指定的工作簿，可以使用下面语句。

```
Workbooks("工作簿名).Save
```

如果需要保存当前活动工作簿，可以使用下面的语句。

```
ActiveWorkbook.Save
```

如果需要保存过程所在的工作簿，可以使用下面的语句。

```
Thisbook.Save
```

【范例 11-5】 保存打开的所有工作簿，并退出 Excel，代码如下所示。

```
Sub 使用 Save 方法保存工作簿()
    MsgBox "下面将要保存所有打开的工作簿，_
    并在保存后退出 Excel2010！"                    '显示提示对话框
    For Each a In Application.Workbooks          '遍历所有打开的工作簿
        a.Save                                   '保存工作簿
    Next
    Application.Quit                             '退出 Excel
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序首先给出提示，如图 11.6 所示。单击“确定”按钮关闭提示对话框后将退出 Excel。

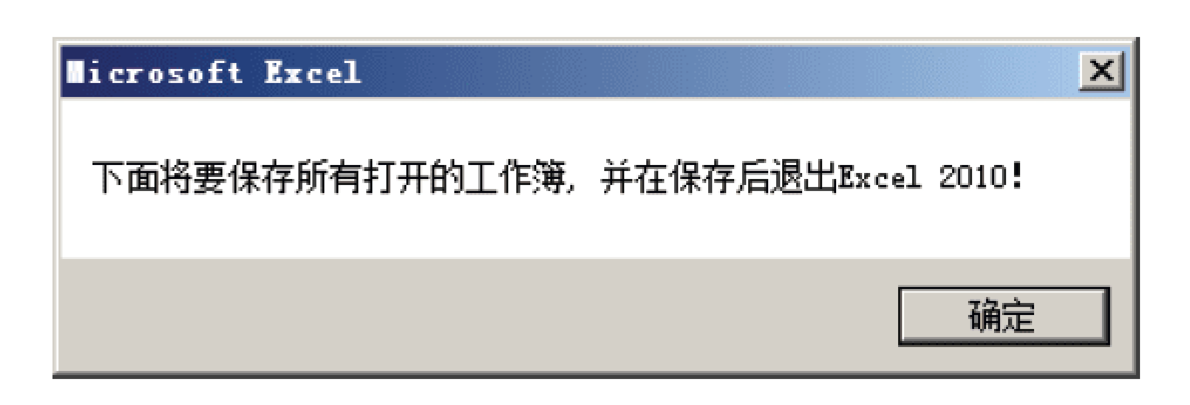


图 11.6　提示对话框

【代码解析】本实例演示在程序退出时，编程保存工作簿的方法。示例使用 For Each…InNext 结构来实现对所有打开工作簿的遍历，使用 Save 方法保存打开的所有工作簿。在完成工作簿的保存后使用 Quit 方法退出 Excel。

注意：使用 Save 方法保存已经保存过的工作簿，如果是第一次保存工作簿，则需要使用 SaveAs 方法。

11.3.2　使用 SaveAs 方法保存 Excel 文档

SaveAs 方法具有比 Save 方法更高的灵活度，在使用该方法保存文档时，能够设置文件保存的位置、文件名以及为文件添加密码等。如，将名为“mybook”工作簿以“mybook1”保存在“d:”盘的“我的 Excel 文档”文件夹中可使用如下代码：

```
Workbooks("myBook").SaveAs "d:\我的 Excel 文档\mybook1.xls"
```

如果需要在保存文件时为文件添加打开密码，可采用如下的代码：

```
Workbooks("myBook").SaveAs "d:\我的 Excel 文档\mybook1.xls" , Password: = 
“123”
```

【范例 11-6】 创建一个新工作簿，并打开“另存为”对话框将该工作簿保存在指定的位置，代码如下所示。

```
01  Sub 使用 SaveAs 方法()
02      Set NewBook = Workbooks.Add                       '添加一个工作簿
03      fName = Application.GetSaveAsFilename(FileFilter: = _
04      "Excel 工作簿(*.xlsx),*.xlsx")                     '获取保存文件路径和文件名
05      If fName <> False Then                            '有文件保存路径和文件名
06          NewBook.SaveAs Filename:=fName                '保存文件
07      End If
08  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序创建一个新的工作簿，然后打开“另存为”对话框。在对话框中选择文件保存文件夹，并输入文档保存文件名，如图 11.7 所示。单击“确定”按钮关闭对话框，新创建的工作簿被保存。

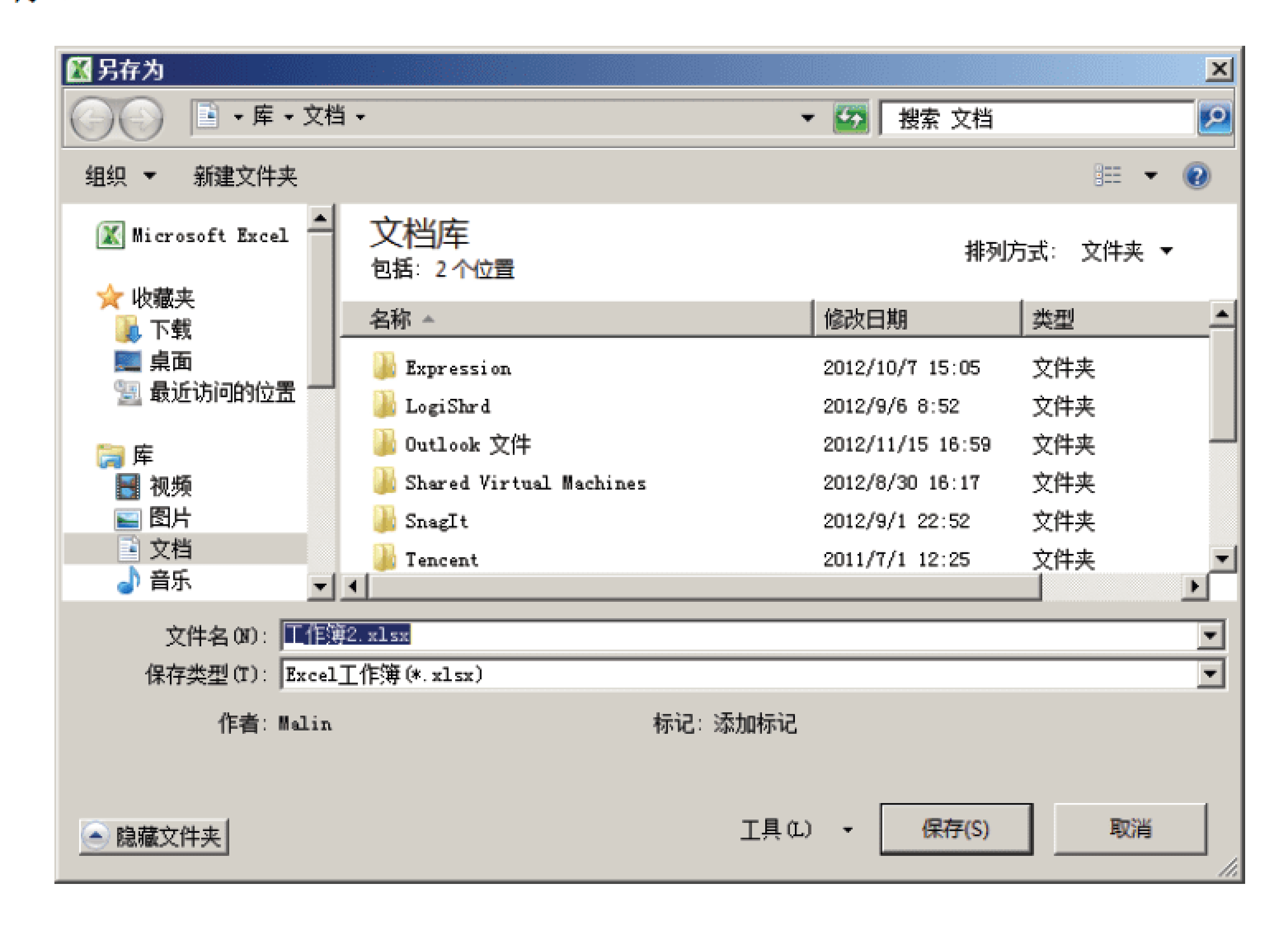


图 11.7　“另存为”对话框

【代码解析】本示例演示了使用 SaveAs 方法保存文档的一般步骤。在本例中，第 02 行代码使用 Add 方法创建一个新的工作簿，并为其指定一个对象变量 NewBook。在第 03 行代码中使用 GetSaveAsFilename 方法打开“另存为”对话框，同时获得用户选择的文档保存路径和文件名。在第 06 行代码使用 SaveAs 方法将创建的新工作簿保存为需要的文件。

注意：*在代码中如果不使用 If 语句来判断“另存为”对话框单击的是哪个按钮，那么当单击“取消”按钮时，文件同样能够保存，不过此时保存的文件名将是“False.xlsx”*

11.4　保护 Excel 工作簿

工作簿的保护包括保护工作簿文件和工作簿元素的保护。保护工作簿文件是设置用户打开工作簿文件的权限，也就是为工作簿添加密码，没有密码的用户无法打开工作簿，即防止非法进入工作簿。而保护工作簿元素是对工作簿结构或窗口进行保护，用户在进入工

作簿后如果没有权限将无法修改工作簿结构和窗口。这两种保护，都可以在 VBA 程序中实现。

11.4.1　设置 Excel 工作簿打开密码

为了保护工作簿，可以为工作簿添加打开权限密码。权限密码可以在 Excel 中直接设置，也可在程序中更改 Workbook 对象的 Password 属性值来设置。在程序中读取 Password 属性值能够获取当前工作簿的权限密码，更改其值能够设置新的密码。

【范例 11-7】 为当前工作簿添加密码，密码使用输入对话框输入，代码如下所示。

```
Sub 设置工作簿打开密码()
    ActiveWorkbook.Password = InputBox("请设置工作簿打开密码！") '输入密码
    MsgBox "你已经设置了工作簿打开密码，文档保存后密码将生效！"'提示密码生效
    ActiveWorkbook.Save                                    '保存文档
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序给出输入对话框要求输入密码，在文本框中输入密码，如图 11.8 所示。输入密码后关闭对话框，程序给出提示信息，如图 11.9 所示。

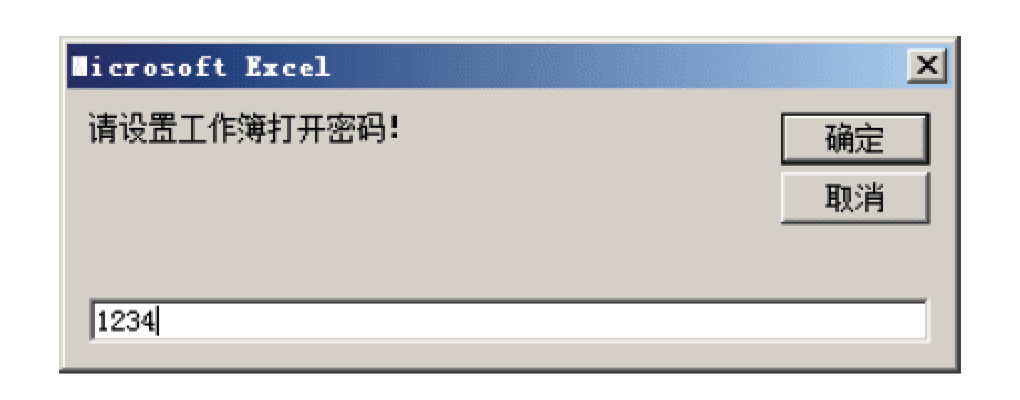


图 11.8　文本输入对话框中输入密码

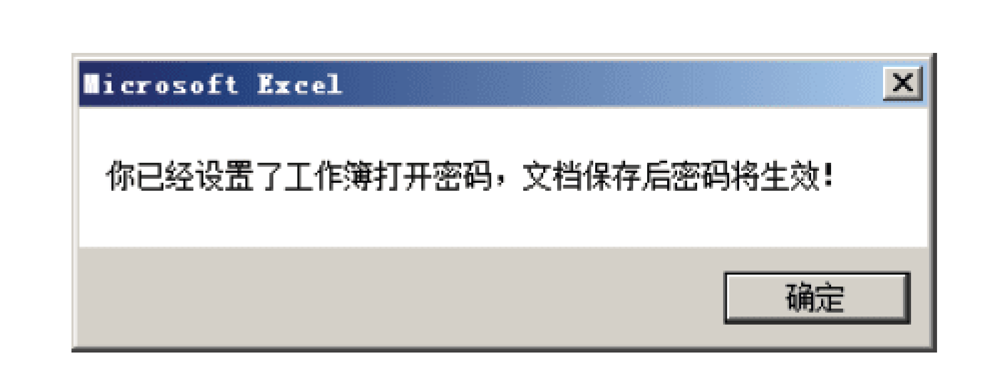


图 11.9　完成密码设置后的提示

保存工作簿后，再次打开文档，文档会给出要求输入打开密码，如图 11.10 所示。正确输入密码，文档将打开，密码输入错误，文档将无法打开。



图 11.10　“密码”对话框

【代码解析】本示例演示编程为文档添加权限密码的方法。在本例中，使用 ActiveWorkbook 获得当前活动工作簿，通过设置工作簿的 Password 属性来设置工作簿的打开密码。

提示： 要取消工作簿的密码，只需将 Password 属性设置为空字符即可，如:

```
Activeworkbook.Password=""
```

11.4.2　设置 Excel 工作簿保护密码

保护工作簿的元素在开发应用系统是十分重要的，可以通过锁定工作表结构来禁止非授权用户对工作簿进行结构上的修改，可以真正起到对工作簿的保护作用。在 VBA 中，使用 WorkBook 对象的 Protect 方法能够实现对工作簿的保护，其语法格式如下：

```
表达式. Protect(Password, Structure, Windows)
```

参数说明如下所示。

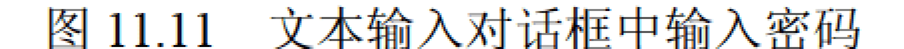
- Password：一个字符串，保护工作簿的密码。
- Structure：如果为 True，则保护工作簿结构。其默认值是 False。
- Windows：如果为 True，则保护工作簿窗口。如果省略此参数，则窗口不受保护。

【范例 11-8】 为当前工作簿添加保护密码，并设置保护工作簿结构和工作簿窗口，代码如下所示。

```
Sub 设置工作簿元素保护密码()
myPassword = InputBox("请设置工作簿权限密码！")          '输入密码
    If myPassword <> False Then                           '是否输入了密码
        ActiveWorkbook.Protect Password:=myPassword, structure:=True,
         _Windows:=True                                   '设置密码并指定保护内容
    End If
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序给出输入对话框要求输入密码，在文本框中输入密码，如图 11.11 所示。此时工作簿的结构和工作簿窗口将受到保护，例如，无法实现改变工作表窗口的大小、无法删除和复制工作簿中的工作表等操作，如图 11.12 所示。




图 11.11　文本输入对话框中输入密码

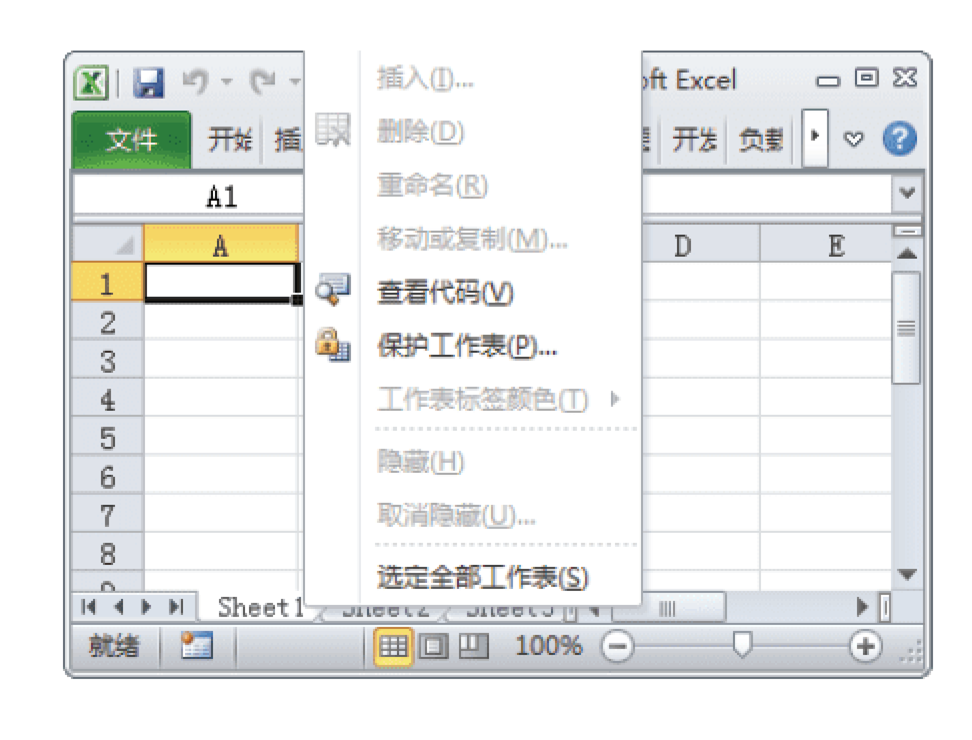


图 11.12　工作表无法删除复制

【代码解析】在本示例中，首先使用 InputBox 函数获取用户的密码输入。使用 If 语句判断是否有密码输入，如果输入了，使用 ActiveWorkbook 的 Protect 方法来设置对工作簿的保护。

提示： 在添加密码后，如果需要取消密码，可使用 UnProtect 方法。这里，必须要指定设置的密码，否则将无法取消当前的密码。如取消本例的密码，可使用下面的语句：

```
Activeworkbook.Unprotect Password:="1234"
```

11.5　使用 Excel 工作簿事件

工作簿事件是 VBA 根据工作簿所发生的动作来执行相应的程序代码。当工作簿被激

活、工作簿中加载了宏或工作簿被打开时，将会引发工作簿事件，执行相应的事件代码。在 Excel 中包括 23 个工作簿事件，几乎涵盖了与工作簿有关的所有变化。使用这些事件，能够编写由事件驱动的程序，实现各种使用操作。

11.5.1　启用或禁用事件

工作簿事件能够触发其他事件，也可触发它自己，这样就有可能会发生某个事件程序运行后又会触发本身这个事件，从而形成一个无休止的循环。要防止这种情况的发生，可以先将事件禁止，然后在程序处理完成后再启动事件。通过设置 Application 对象的 EnableEvents 属性值，可以实现对事件的禁用和启用。当将其值设置为 True 时，将启用对象事件，其值设置为 False 时将禁用事件。

【范例 11-9】当工作表中数据发生改变，提示被内容被修改的内容，同时更改单元格填充色以示提示，代码如下所示。

```
01  Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)
02      MsgBox "单元格中数据被改为" & Target.Value              '显示提示
03      Application.EnableEvents = False                        '禁止事件
04      Target.Interior.ColorIndex = 3                          '更改单元格颜色
05      Cells(Target.Row, 6) = "单元格" & Target.Address &"内容被修改"'记录修改地址
06      Application.EnableEvents = True                         '启用事件
07  End Sub
```

【运行结果】在 VBA 编辑器中打开工程管理器，双击其中的“ThisWorkBook”选项打开“代码”窗口，在代码窗口中输入上述代码。切换到 Excel，当工作表中某个单元格数据改变时，给出提示对话框，如图 11.13 所示。关闭对话框后，单元格 A1 中输入刚才输入的值，如图 11.14 所示。

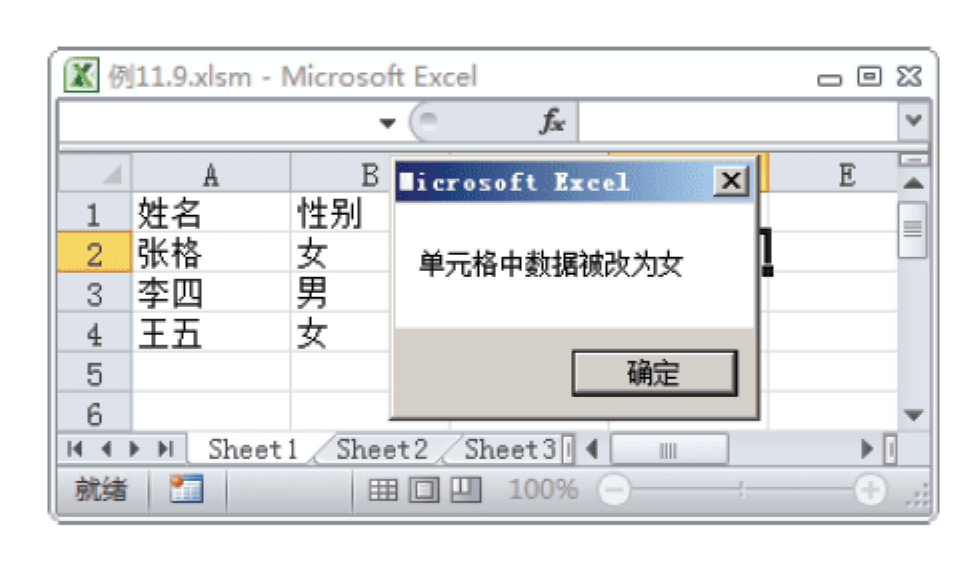


图 11.13　提示对话框

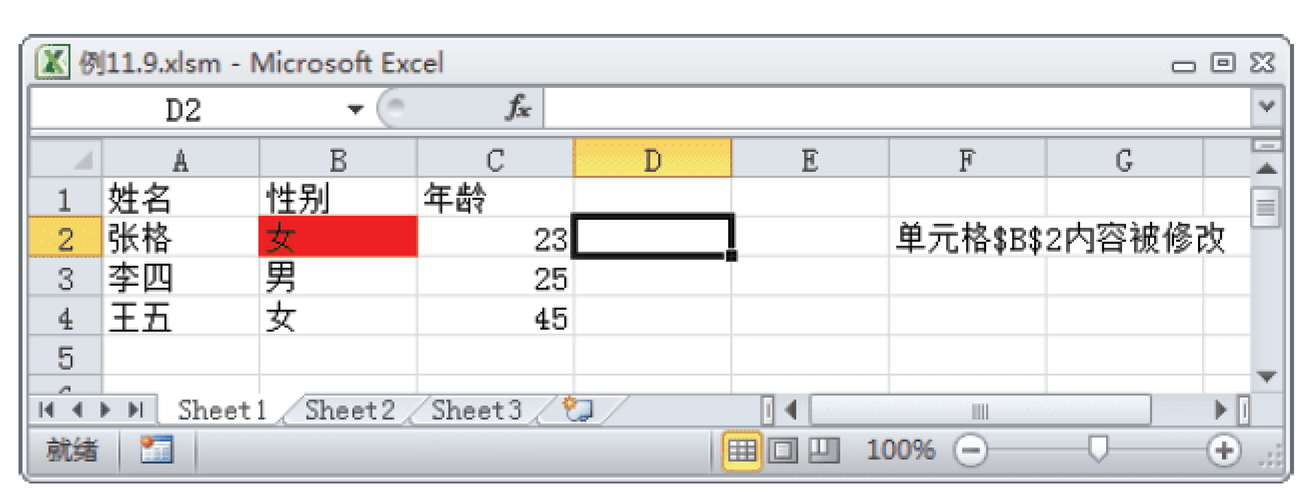


图 11.14　被修改的单元格被填充颜色

【代码解析】本示例演示使用 EnableEvents 来启动和禁止事件的方法。在程序中，SheetChange 事件是在工作表发生变化时触发的事件。在本程序中，第 03 行语句禁用事件，这样可以避免程序反复触发 SheetChange 事件而进入死循环。在完成向单元格内容更改后的第 06 行，重新启动事件，以保证在新的事件发生时能够执行事件响应程序。

注意：事件是允许带参数的，这些参数能够将值传给过程。本示例中，就是使用事件的 Target 参数获得了修改内容的单元格对象。

11.5.2　使用 Excel 工作簿的窗口大小更改事件

在 Excel 中，调整工作簿窗口大小时将会触发 WindowSize 事件。该事件带有一个参数 Wn，其能够在事件发生时返回一个 Window 对象，使用该对象能够实现对窗口的控制。下面通过范例介绍 WindowSize 事件的使用。

【**范例 11-10**】调整工作簿窗口大小后，显示当前工作簿窗口的大小，代码如下所示。

```
01  Private Sub Workbook_WindowResize(ByVal Wn As Window)
02      a = Wn.Caption                              '获得窗口的标题
03      w=Wn.Width                                  '获得窗口宽度
04      h = Wn.Height                               '获得窗口高度
05      t = Wn.Top                                  '获得窗口上端在屏幕中的位置
06      l = Wn.Left                                 '获得窗口左侧在屏幕中的位置
07      MsgBox "当前工作簿窗口为：" & a & Chr(13) _
08      & "当前窗口的宽度为：" & w & "像素" & Chr(13)
09      & "高度为：" & h & "像素" & Chr(13) _
10      & "当前窗口在屏幕上位置为：(" & t & "，" & l & ")" _
11      , vbOKOnly, "工作簿窗口信息"                '显示窗口的属性信息
12  End Sub
```

【运行结果】在 VBA 编辑器中打开工程管理器，双击其中的“ThisWorkBook”选项打开“代码”窗口，在代码窗口中输入上述代码。切换到 Excel，拖动当前工作簿窗口边框改变其大小，事件被触发，程序给出提示，显示当前窗口的名称、宽度、高度和在屏幕中的位置，如图 11.15 所示。

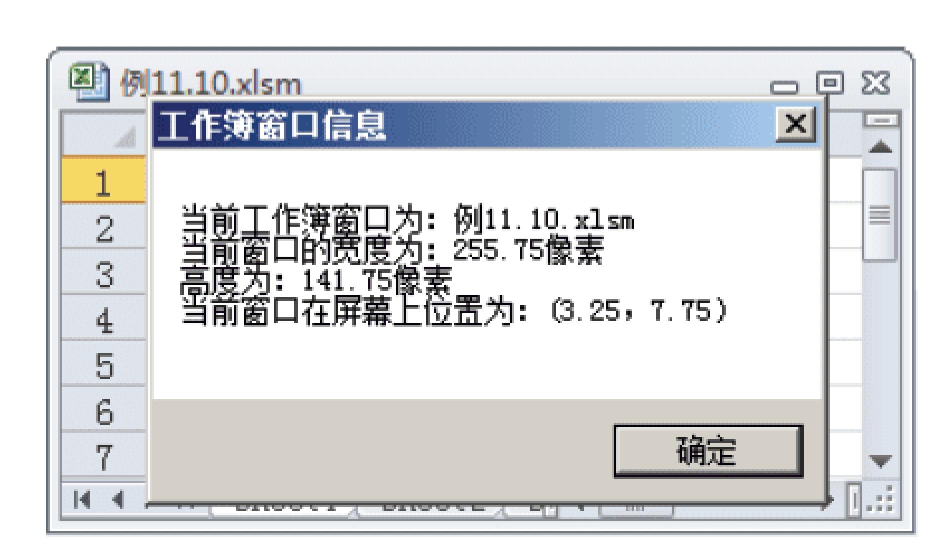


图 11.15　提示对话框显示窗口的信息

【代码解析】本段代码演示 WindowResize 事件的响应过程。事件的 Wn 参数返回 Window 对象，该对象代表当前窗口。第 02～06 行代码就是分别获取这个 Windows 对象的标题、宽度和高度等属性值。在代码的第 07～11 行使用 MsgBox 函数分行显示需要获得的窗口信息。

🔔**提示**：在 VBA 中，屏幕坐标系的坐标原点位于屏幕的左上角，以向右方向为横轴的正方向，向下的方向为纵轴的正方向。

11.5.3　使用 Excel 工作簿的打开事件

Workbook 的 Open 事件是在工作簿被打开时触发，通过编写该事件的响应程序可以使程序随工作簿启动而自动运行。由于该事件在工作簿打开后不可能再次被触发，因此事件代码只能是那些只需要执行一次的代码。在 VBA 中常使用 Open 事件代码来实现程序的初始化操作，如初始化控件、设置窗口的大小以及设置各个变量的初始值等。

【**范例 11-11**】打开工作簿，并在工作簿中记录打开时间，代码如下所示。

```
Private Sub Workbook_Open()
    Application.WindowState = xlMaximized           '应用程序窗口最大化
    Sheets("sheet1").Cells(Range("a1048576"). _
    End(xlUp).Row + 1, 1).Value = "文档打开的时间为：" & Now
                                                    '将当前时间填入单元格
End Sub
```

【运行结果】在 VBA 编辑器中打开工程管理器，双击其中的“ThisWorkBook”选项打开“代码”窗口，在代码窗口中输入上述代码，保存文件后关闭工作簿。再次启动工作簿，Excel 程序窗口将自动最大化，同时在姓名列的最后写上打开工作簿的时间，如图 11.16 所示。

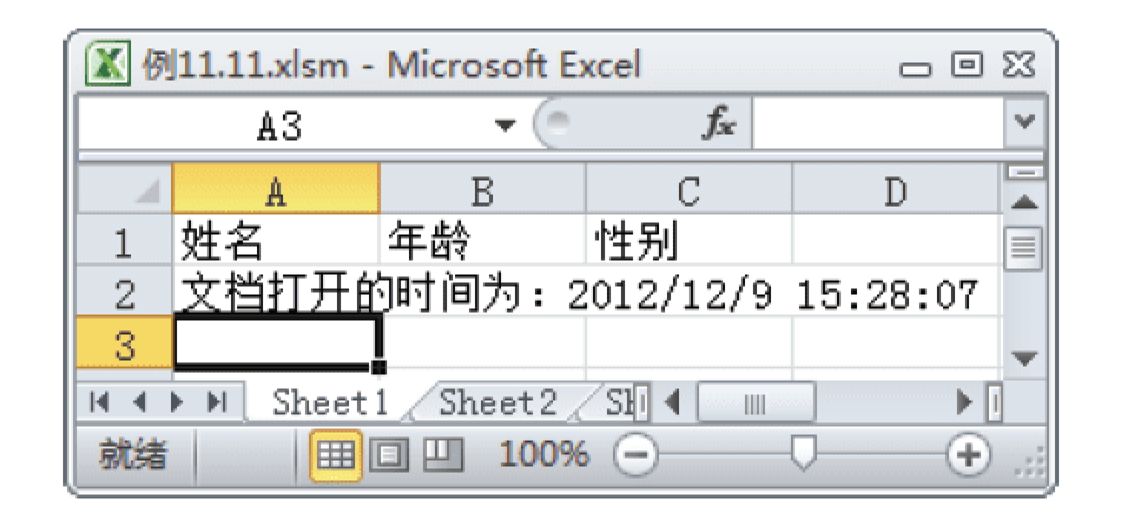


图 11.16　单元格显示工作簿打开时间

【代码解析】在本示例中，工作簿的 Open 事件响应代码完成两个工作，设置 Application 对象的 WindowState 属性值以及使 Excel 程序窗口最大化。使用 Now 函数获得当前时间，并将时间值写入姓名列的最后。

注意：工作簿的 Open 事件代码有可能不被再执行，如在打开工作簿时如果用户按住 Shift 键就可以跳过该事件代码。

11.5.4　使用 Excel 工作簿的工作表激活事件

SheetActivate 事件是在有工作表被激活时发生，使用该事件能够在激活工作表时对工作表的各个元素进行设置，也可用于执行需要打开工作表运行的程序。

【范例 11-12】 打开工作表，自动选择 A1 单元格，同时提示选择工作表的名称，代码如下所示。

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    Range("A1").Select                          '选择第一个单元格
    MsgBox "当前激活的工作表是：" & Sh.Name     '显示被激活工作表的名称
End Sub
```

【运行结果】在 VBA 编辑器中打开工程管理器，双击其中的“ThisWorkBook”选项打开“代码”窗口，在“代码”窗口中输入上述代码。切换到 Excel 窗口，选择一个工作表，程序给出提示，同时自动选择 A1 单元格，如图 11.17 所示。

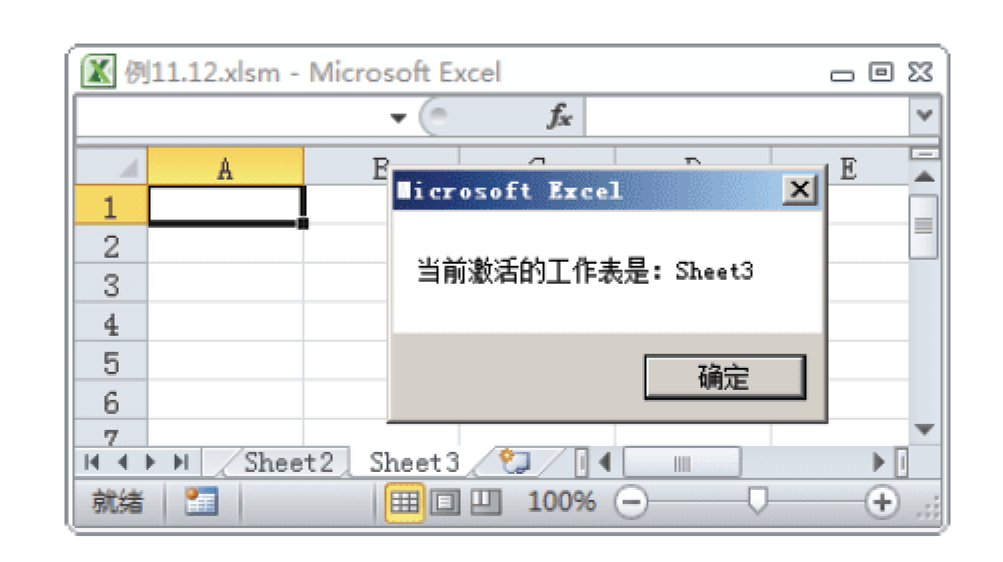


图 11.17　显示提示信息并自动选择 A1 单元格

【代码解析】本示例演示 SheetActivate 事件的使用。在程序中，使用 SheetActivate 事件来实现自动选择单元格并显示提示信息。事件参数 Sh 返回被激活工作表，在代码的第 03 行使用 Sh.Name 语句获得激活工作表的名称。

提示：SheetActivate 事件带有参数 Sh，该参数表示激活的工作表对象。使用该参数能够在程序中读取对象的属性和使用对象的方法。

11.5.5　使用 Excel 工作簿的关闭之前事件

在 Excel 中，关闭工作簿之前，会产生 BeforeClosed 事件。如果工作簿已经被更改，该事件发生时提示用户是否保存更改。通过编写 BeforeClose 事件程序，能够在工作簿关闭前完成一些必要工作，如关闭窗体、复位工作表中的数据或将某些修改的参数还原等。

【范例 11-13】 关闭工作簿，如果文档没有保存，给出提示要求用户保存，代码如下所示。

```
01  Private Sub Workbook_BeforeClose(Cancel As Boolean)
02      If Me.Saved = False Then                          '判断文档是否保存
03          a = MsgBox("工作簿将要关闭，但尚未保存， _
04          是否保存?", vbOKCancel, "提示")                 '显示提示信息
05          If a = vbOK Then                              '单击了“确定”按钮
06              Me.Save                                   '保存文档
07          Else
08              Cancel = True                             '停止关闭操作
09          End If
10      End If
11  End Sub
```

【运行结果】在 VBA 编辑器中打开工程管理器，双击其中的“ThisWorkBook”选项打开“代码”窗口，在“代码”窗口中输入上述代码。切换到 Excel 窗口，单击工作簿窗口上的“关闭”按钮执行关闭操作时触发事件。此时，如果当前工作簿没有保存，则程序给出提示对话框，如图 11.18 所示。单击“确定”按钮程序将保存当前工作簿，单击“取消”按钮程序将退出。



图 11.18　显示提示信息

【代码解析】在本示例中，第 02 行代码判断当前工作簿是否已经保存，如果保存则给出提示对话框。第 04 行代码判断提示对话框是否单击了“确定”按钮，如果是，则执行第 06 行代码执行保存操作。如果没有则将事件的 Cancel 参数设置为 True，停止关闭工作簿的操作。

提示：Cancel 参数在 BeforeClose 事件过程中很重要，其值决定了是否执行关闭工作簿的操作。这里，如果不将 Cancel 参数设置为 True 取消关闭操作，则由于文档没有保存，在本范例关闭提示对话框后，将得到 Excel 的提示对话框，提示是否保存当前未保存的文件。

11.6　小　　结

本章介绍了工作簿对象的使用方法，通过本章的介绍读者将会学习工作簿对象的常用属性、方法和事件，能够通过编程实现工作簿的新建、打开、保存以及工作簿的保护等操作，熟悉 VBA 事件编程的特点，并能够使用常见事件解决常见的问题。

事件实际上是一种能够被对象识别的动作，通过在事件过程中放置代码来响应这些动作。使用事件编程能够为程序的编写提供更大的灵活性，但随着事件的增多，管理事件间的交互将变得十分重要。为此，可以使用 Application 对象的 EnableEvents 属性来启用或禁用事件。

11.7　本 章 习 题

1．下面哪个语句无法正确实现对工作簿的引用？（　　）

A．ActiveWorkbook　　B．Workbook(2)

C．Workbook(Workbooks.Count)　　D．Workbook(1)

2．需要打开名为 D:\123 文件夹下的名为“学生成绩表.xls”文档，且文档带有打开密码为“123”，应该使用下面哪个语句？（　　）

A．Open filename:="D:\123\book1.xls",password:="123"

B．Workbooks.Open filename:="D:\123\book1.xls"

C．Workbooks.Open filename:="D:\123\book1.xls",password:="123"

D．Workbooks.Open filename:="D:\123\book1.xls","123"

3．运行下面代码，对运行结果的描述错误的是哪一项？（　　）

```
Sub test()
    ActiveWorkbook.Protect Password:="password", structure:=True,
     Windows:=True
End Sub
```

A．工作簿中的工作表无法删除　　B．工作簿的工作表窗口无法更改大小

C．工作簿的工作表的单元格无法调整大小　D．工作簿的工作表无法更名

4．当工作表中某个单元格数据被修改时，选择下面程序的运行结果。（　　）

```
Private Sub Workbook SheetChange(ByVal Sh As Object, ByVal Target As
Range)
    Cells(1,1) = "单元格内容被修改！"
End Sub
```

A．单元格内容被修改但 A1 单元格为空

B．程序进入死循环

C．工作表的 A1 单元格写入“单元格内容被修改！”

D．Excel 将自动退出

5．设计一个程序，当打开多个工作簿时，能够在工作簿中查找命名“工作表”的工作表。

【提示】本程序可以通过两层循环嵌套来实现，外层循环保证遍历打开的所有工作簿，内层循环遍历每一个工作簿的工作表。循环体中判断获得的工作表名是否是“工资表”，根据查询的结果给出不同的提示。

6．编写程序，程序能够判断用户在输入对话框中指定的工作簿是否打开。

【提示】使用 InputBox 函数获得用户输入工作簿名，使用循环结构遍历所有打开的工作簿，判断每一个工作簿名是否是输入的名称。如果是，提示工作簿打开，否则，提示工作簿未打开。

第 12 章　工作表对象

WorkSheet 对象表示 Excel 的工作表，通过使用它可以引用 Excel 文档中的工作表、新建工作表、删除工作表、隐藏工作表、复制工作表、移动工作表、打印工作表，以及使用工作表事件来响应对工作所做的激活、更改及右击操作。主要的内容和学习目的如下：

- 使用工作表对象引用 Excel 工作表的方法；
- 使用工作表对象新建和删除 Excel 工作表的方法；
- 使用工作表对象选取和隐藏 Excel 工作表的方法；
- 使用工作表对象复制和移动 Excel 工作表方法；
- 使用工作表对象打印 Excel 工作表的方法；
- 通过工作表对象响应 Excel 工作表常用事件。

12.1　引用 Excel 工作表对象

Excel VBA 包含 WorkSheets 和 Sheets 这两个工作表对象集合。Sheets 对象集合包括 WorkSheet 对象和 Chart 对象（即图表对象）。WorkSheets 对象集合包含 Worksheet 对象。在程序中要实现对工作表的操作，首先需要引用工作表对象，即指定操作的工作表对象。本节将首先介绍在 Excel VBA 中引用工作表的方法。

12.1.1　使用名称引用 Excel 工作表

在 Excel VBA 中，工作表都有两个名称，一个是工作表代码名称，另一个是工作表名称。代码名称是在程序代码中引用的名称，工作表名称则是工作表在程序中使用 Name 获得的属性值，实际上这个名称就是在 Excel 工作表标签上显示的名称。在 Visual Basic 编辑器中，工程资源管理器的对象列表中同时列出来工作表的这两个名称，如图 12.1 所示。

例如，对于第一个工作表，Sheet1 就是第一个工作表的代码名称，而其后括号中的“成绩表”就是这个工作表的工作表名称。这个名称与 Excel 界面中工作表标签上的名称一致，如图 12.2 所示。

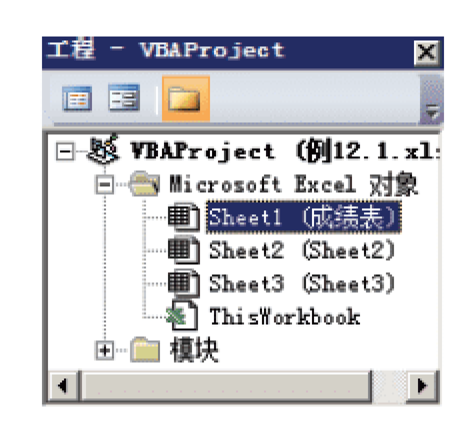


图 12.1　资源管理器中显示的工作表名称

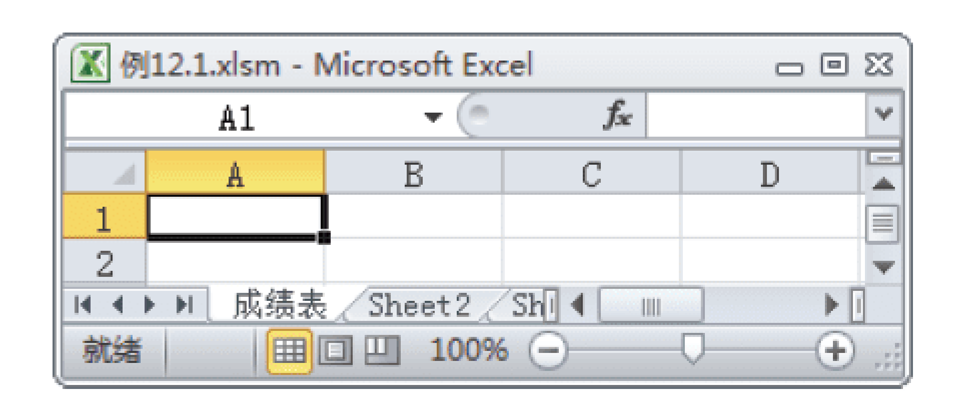


图 12.2　Excel 中的标签名

在 VBA 中，对工作表的引用，可以使用工作表名称来实现，下面通过一个实例来介绍这种引用方法的使用。

【范例 12-1】 编程实现对图 12.2 中工作表的引用，代码如下所示。

```
01  Sub 使用名称引用工作表()
02      Debug.Print Sheet1.Name                       '使用工作表代码名称
03      Debug.Print Sheet2.Name
04      Debug.Print Worksheets("成绩表").Name          '使用工作表名
05      Debug.Print Sheets("sheet3").Name
06      Debug.Print Sheets("成绩表").Name
07  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，在“立即窗口”显示不同方法引用工作表对象所获得的 Name 属性的值，如图 12.3 所示。

【代码解析】本示例使用不同方法来实现工作表的引用。第 02 行和第 03 行代码直接使用工作表的代码名称来引用。第 04~06 行代码使用 Worksheets 和 Sheets 对象名结合工作表名来实现对工作表的引用。

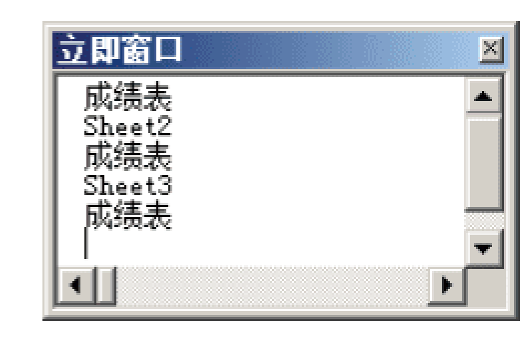


图 12.3　程序运行效果

提示：工作表对象名是创建工作表时自动获得的，可以在 Visual Basic 编辑器的“属性”窗口中对其进行修改。

12.1.2　使用索引号引用 Excel 工作表

在 Excel VBA 中，可以使用工作表索引号来表示和引用工作表。工作表索引号表示该工作表在工作簿标签栏上的位置，索引号从工作表标签的最左边开始向右依次计数，最左边的工作表是第 1 个工作表，向右依次类推。下面通过一个范例来熟悉索引号引用工作表的方法。

【范例 12-2】 使用索引号实现对图 12.4 中工作表的引用，代码如下所示。

```
01  Sub 使用索引号引用工作表()
02      Debug.Print Worksheets(1).Name                      '直接使用索引号
03      Debug.Print Sheets(1).Name
04      Debug.Print Worksheets(Worksheets.Count).Name        '引用最后一个工作表
05      Debug.Print Sheets(ActiveSheet.Index).Name           '引用当前工作表
06  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，在“立即窗口”显示不同方法引用工作表对象所获得的 Name 属性的值，如图 12.4 所示。

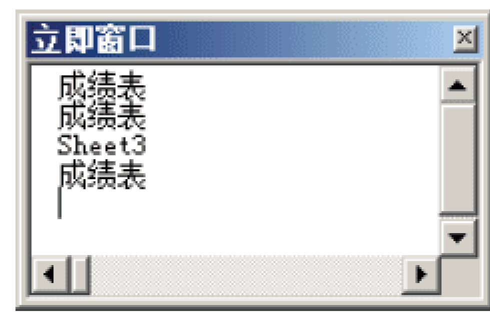


图 12.4　程序运行效果

【代码解析】本范例演示使用索引号来引用工作表的方法。代码第 02 行和第 03 行是直接使用索引号来实现对工作表的引用。Worksheets 对象的 Count 属性值是当前工作簿中工作表的数量，在第 04 行中使用该值作为索引号引用最后一个工作表。WorkSheet 对象

的 Index 属性值是对象的索引号，在程序的第 05 行使用当前激活工作表的索引号来引用工作表。

提示：在编程时，使用工作表名使程序通俗易懂。而当在编程中需要使用多个工作表时，可以使用变量来替代索引号，并且能够通过循环结构来处理多张工作表，这样能够使程序更为简捷。

12.2　新建和删除 Excel 工作表

在默认状态下，Excel 会自动生成 3 个工作表，根据需要用户可在文档中增添新的工作表或删除已有的工作表。在 VBA 程序中，通过编写程序代码同样可以实现创建新工作表和删除已有工作表的操作。

12.2.1　新建 Excel 工作表

Sheets 对象和 Worksheets 对象都有 Add 方法，使用该方法能够创建新的工作表。Add 方法的语法结构如下：

```
表达式.Add(Before, After, Count, Type)
```

参数说明如下所示。

- Before：指定一个工作表对象，新建的工作表将置于此工作表之前。
- After：指定工作表的对象，新建的工作表将置于此工作表之后。
- Count：要添加的工作表数，其默认值为 1。
- Type：指定新建工作表的类型。可以为下列 XlSheetType 常量之一，xlWorksheet、xlChart、xlExcel4MacroSheet 或 xlExcel4IntlMacroSheet。如果基于现有模板插入工作表，则指定该模板的路径。默认值为 xlWorksheet。

在使用 Add 方法时，如果新建一个空白工作表，可以使用下面语句：

```
Worksheets.Add
```

创建工作表时，可以使用 Count 参数来指定新增工作表的数量，例如当需要再增加 4 个工作表时，可以使用下面的语句：

```
WorkSheets.Add Count:=4
```

使用 After 和 Before 参数可以指定创建工作表的位置。如在最后一个工作表后再添加一个工作表，可以使用如下的语句：

```
Sheets.Add After:=Sheets(Sheets.Count)
```

【范例 12-3】 新建 3 个工作表，并为这些工作表命名，代码如下所示。

```
Sub 新建工作表()
    Dim sht As Worksheet, i As Integer                  '变量声明
```

```
03        For i = 1 To 3                              '进行循环操作
04            Set sht = Sheets.Add                    '新建工作表
05            sht.Name = "七年级" & "(" & i & ")班成绩表"  '工作表命名
06        Next
07    End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，工作簿中增加 3 个工作表，如图 12.5 所示。

【代码解析】本示例演示同时创建多个工作表的方法。在本示例中，代码的第 02 行声明一个 Worksheet 对象变量 sht，在第 04 行将新建的工作表指定给这个对象变量，然后使用对象变量来实现对象的引用，通过设置工作表对象的 Name 属性值来命名新建的工作表。

提示：在默认情况下，新建工作表后，Excel 会根据当前工作簿中最大工作表标签号来给新建工作表命名。新工作表名在默认情况下是根据工作表创建的顺序，最大标签号加 1。使用工作表的 Name 属性可以更改这个名称。

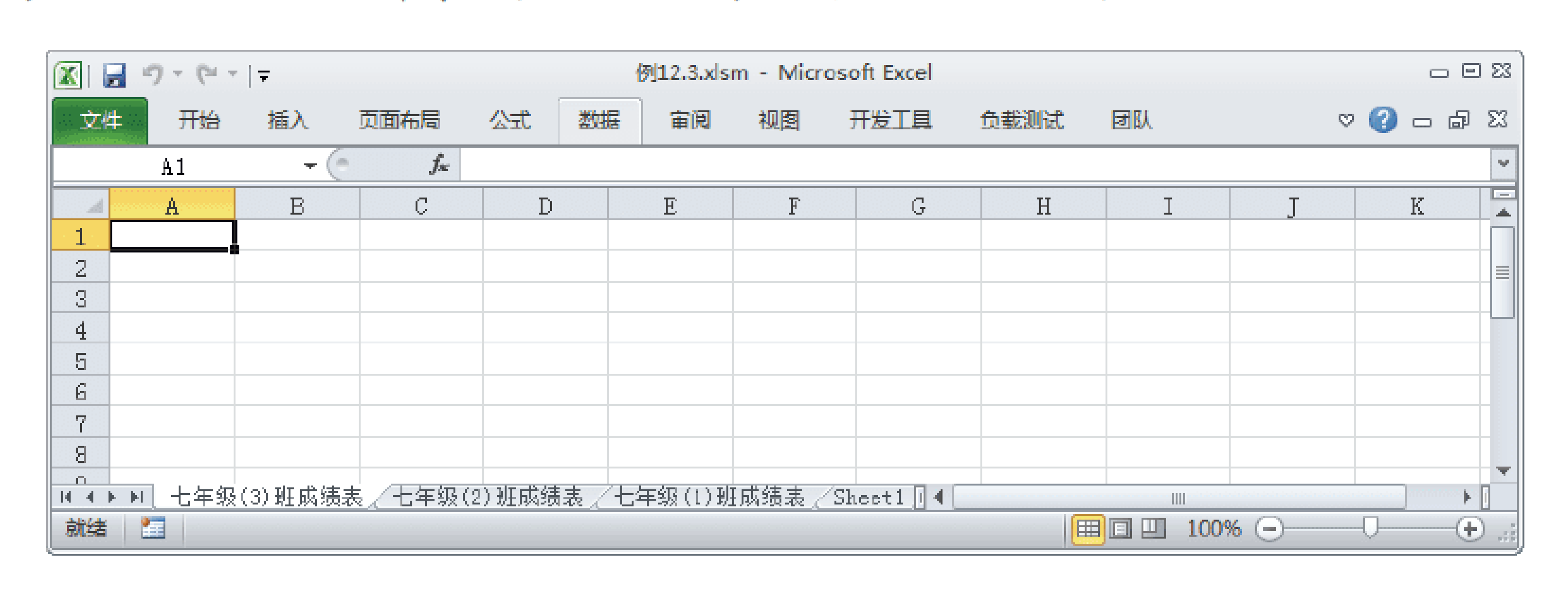


图 12.5　程序运行效果

12.2.2　删除 Excel 工作表

对工作表的操作，有时需要删除已经存在的工作表。删除工作簿中的工作表，可以使用 WorkSheet 对象的 Delete 方法来实现。该方法在使用时显示一个对话框，用于提示用户确认是否删除。如果用户在对话框中单击“取消”按钮，则返回 False。如果用户单击“删除”按钮，则返回 True。该方法的语法结构如下：

```
工作表名.Delete
```

【范例 12-4】 删除范例 12-4 文件中的“sheet1”、“sheet2”和“sheet3”，代码如下所示。

```
Sub 删除工作表()
    Dim sht As Worksheet, n As Integer
    n = 1                                   '变量初始化
    For Each sht In Sheets                  '遍历工作簿中所有工作表
        If sht.Name = "Sheet" & n Then      '如果工作表名是“Sheet”+数字
            sht.Delete                      '删除工作表
            n = n + 1                       '变量加 1
        End If
    Next
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，Excel 会给出对话框提示删除工作表，如图 12.6 所示。单击“确定”按钮即可将工作表删除。这样的提示对话框会出现 3 次，完全删除以“Sheet”开头的工作表后的工作簿，如图 12.7 所示。



图 12.6　Excel 提示对话框

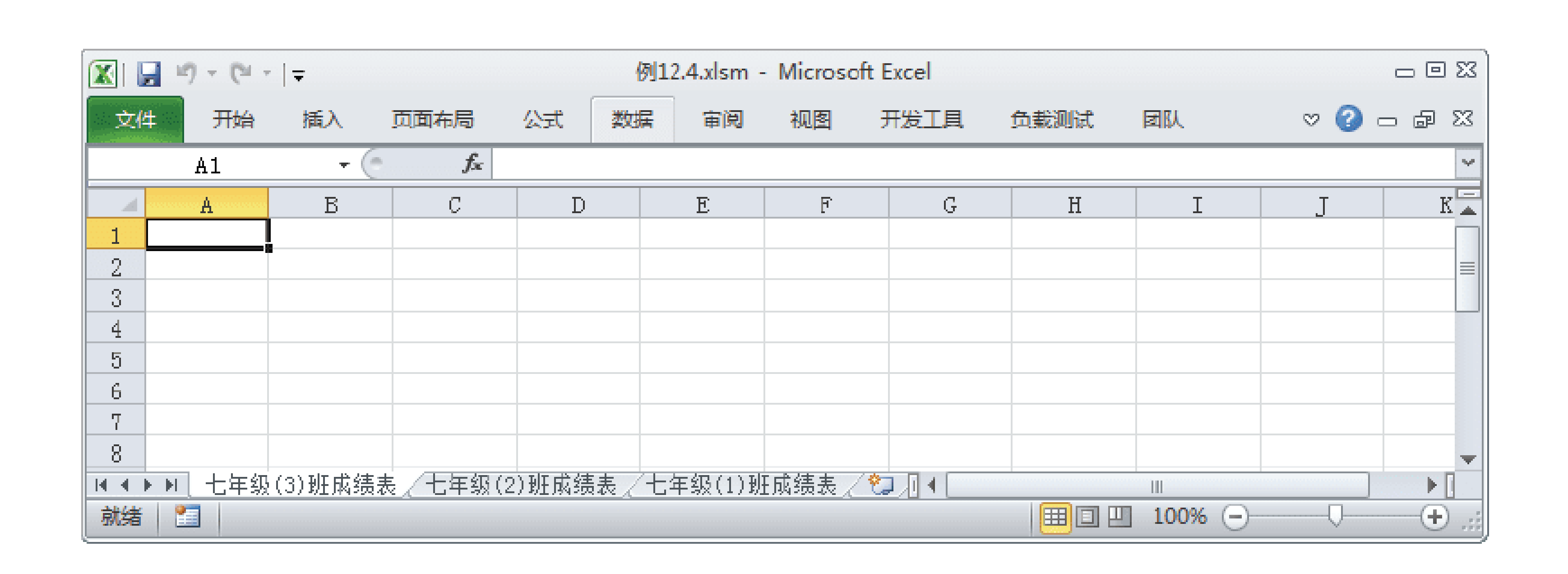


图 12.7　删除工作表后的效果

【代码解析】在本范例中，使用 For Each …In Next 来实现对工作簿中所有工作表的遍历。使用 If 结构以工作表名作为条件来判断工作表是否是为需要删除的工作表，如果是，使用 Delete 方法来删除该工作表。在代码中，变量 n 用于计数，作为名为“Sheet”工作表名中的标号来实现对工作表的引用。

警告：引用工作表时，要注意工作表名的大小写。这里，如果第 05 行“Sheet”如果没有大写，将无法实现对名为“Sheet”的工作表的操作。

12.3　选取和隐藏 Excel 工作表

在进行工作表的操作时，首选需要选择工作表，使用 Select 方法是实现工作表选取的首选方法。在进行数据管理时，为了保护某些数据，可以对这些数据进行隐藏，这也就是隐藏工作表的意义。本节将介绍使用 VBA 选择和隐藏工作表的方法。

12.3.1　选择 Excel 工作表

要选择工作簿中的工作表，可以使用 Select 方法来实现。例如，选择工作表名或工作表代码名均为默认的“Sheet1”的工作表时，可以使用下面的语句：

```
Sheet1.Select
```

或

```
Sheets("Sheet1").Select
```

如果需要激活工作表，可以使用 Activate 方法来实现，激活工作表实际上相当于对工作表进行了选择。例如，激活工作表名为“Sheet1”的工作表可以使用如下语句：

```
Sheets("Sheet1").Activate
```

🔔**注意：** Select 方法和 Activate 方法在使用上是有区别的，使用 Select 方法能够一次选择多个工作表，但只能有一个工作表被激活。

【范例 12-5】 在默认的工作簿中同时选择第一个和最后一个工作表，并在选择工作表的 A1 单元格中输入当前选择工作表名，代码如下所示。

```
01  Sub 选取工作表()
02      Dim p As String                                  '声明变量
03      Sheets(Array(1, 3)).Select                       '选择工作表
04      For Each r In Workbooks(1).Windows(1).SelectedSheets
                                                         '遍历所有选择工作表
05          p = r.Name & Chr(13) & p                     '连接工作表名
06      Next
07      MsgBox "当前选择的工作表为" & Chr(13) & p          '显示打开的工作表
08  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，工作簿中的第 1 个和第 3 个工作表被选择，程序给出提示对话框，显示被选择工作表名，如图 12.8 所示。

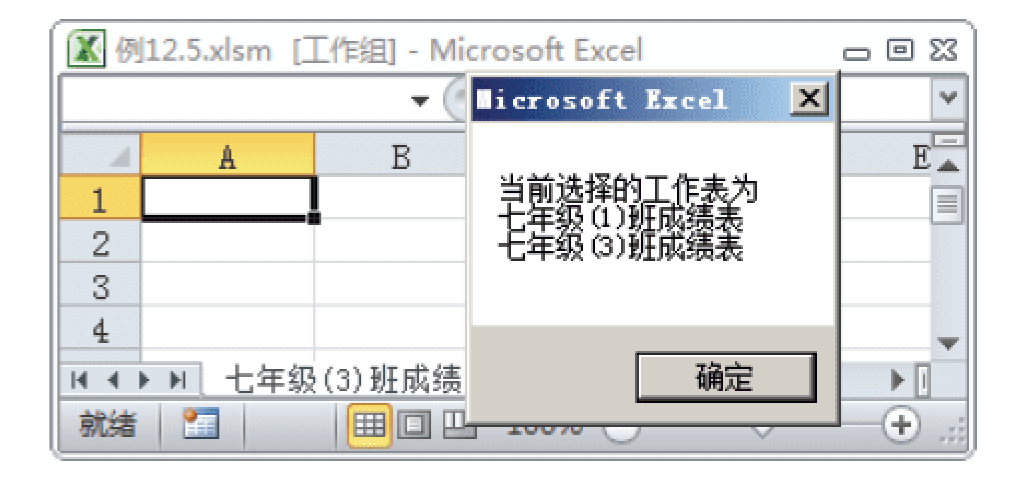


图 12.8　工作表被选择并显示工作表名

【代码解析】在本范例演示同时选择多个工作表的方法。程序的第 02 行代码使用 Array 函数来同时选择多个数据表要获得选择的工作表，需要使用 Windows 对象的 SelectedSheets 属性，该属性值指定了当前窗口中选择的工作表。程序的第 04~06 行通过 For…Next 循环来遍历所有选择的工作表，在循环体中将工作表名拼合为一个字符串。

🔔**提示：** 在使用 Array 数组函数选择多个工作表时，函数的参数可以是工作表索引号，也可以是工作表名，参数间要用逗号“,”连接。

12.3.2　隐藏 Excel 工作表

在工作簿中，为了保护某些包含重要数据的工作表，可以将其隐藏以避免工作表中数据被随意篡改。在 Excel VBA 中，通过设置 Worksheet 对象的 Visible 属性的 xlSheetVisibility 值来确定对象是否可见。

当 xlSheetVisibility 值设置为 xlSheetHidden 时，将隐藏工作表。对于隐藏的工作表用户可以通过菜单取消隐藏。当其值设置为 xlSheetVeryHidden 时，将隐藏对象。如果需要使

对象重新可见，可以将此属性设置为 True 即可。当其值设置为 xlSheetVisible 时，工作表将显示。

提示： 改变工作表的 Visible 属性值，既可以使用 Excel 常量（如 xlSheetHidden），也可以使用数字值（如 0，即 xlSheetHidden）。当前工作表被隐藏后，其后面一个工作表将自动成为活动工作表。

【范例 12-6】 创建一个应用程序由用户决定隐藏哪个工作表，如果选择的工作表已经隐藏，则取消其隐藏，代码如下所示。

```
Sub 工作表隐藏示例()
    a = InputBox("输入需要隐藏的工作表索引号,如果该工作表已经隐藏，则该工 _
    作表将取消隐藏")                                  '输入索引号
    On Error GoTo check                              '数据输入错误转到错误提示
    If Sheets(Val(a)).Visible <> 0 Then              '如果工作表没有隐藏
        Sheets(Val(a)).Visible = xlSheetHidden            '隐藏工作表
    Else
        Sheets(Val(a)).Visible = -1                  '工作表隐藏时，使其可见
    End If
check:
End Sub
```

【运行结果】启动 Excel，创建一个工作簿，工作簿默认有 3 个工作表，如图 12.9 所示。创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序给出输入对话框要求输入索引号，如图 12.10 所示。输入索引号后单击“确定”按钮关闭输入对话框，指定的工作表被隐藏，如图 12.11 所示。如果指定的工作表已经处于隐藏状态，则输入索引号后该工作表将取消隐藏。

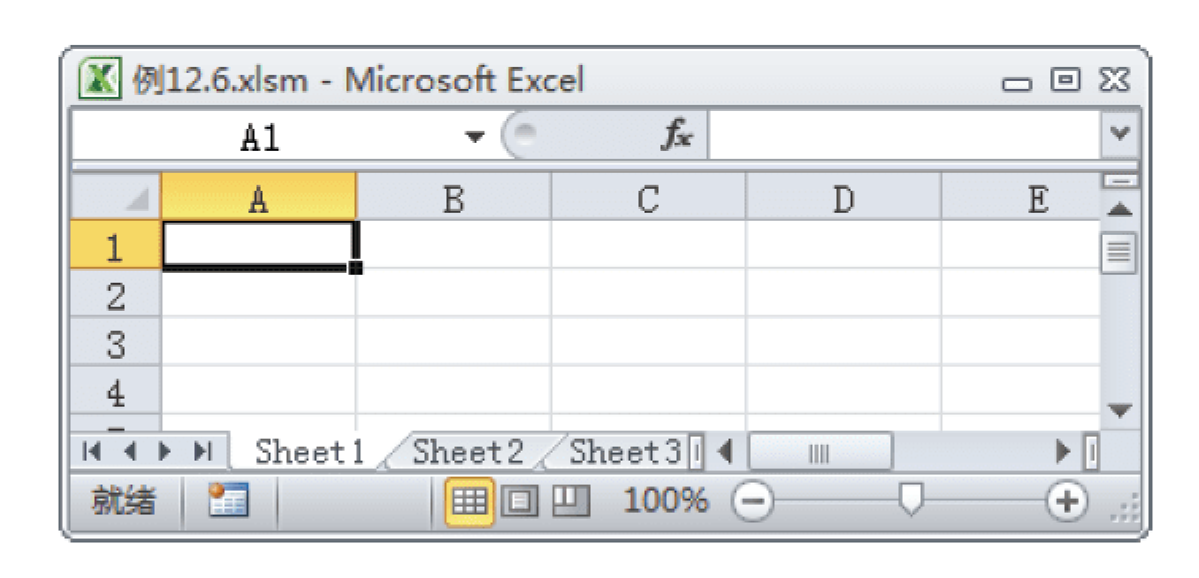


图 12.9　有 3 个工作表的工作簿

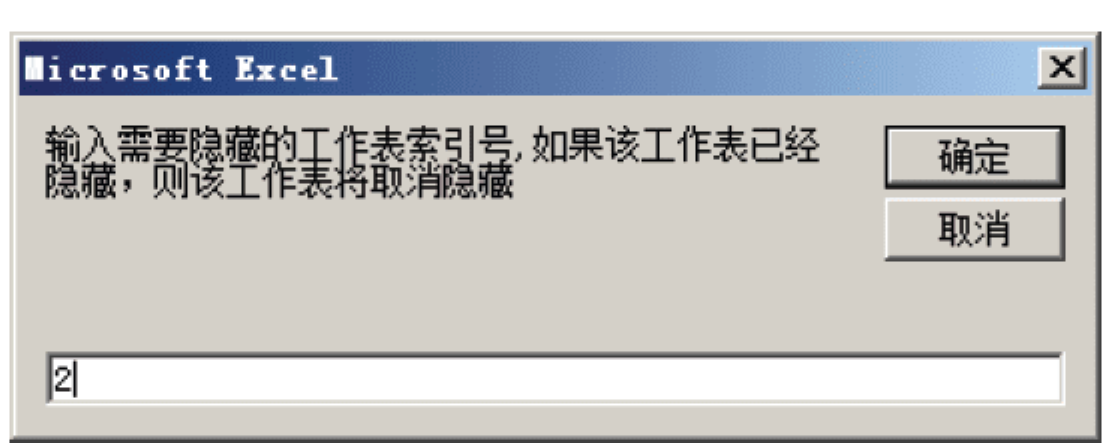


图 12.10　程序给出输入对话框

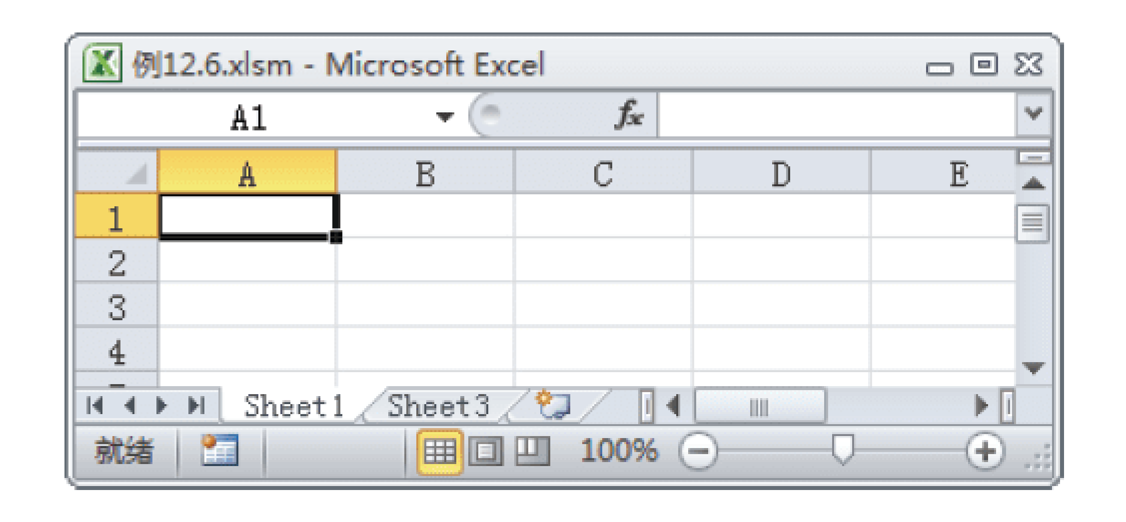


图 12.11　程序运行前后效果对比

【代码解析】本示例演示编程隐藏指定工作表的方法。在代码中使用 InputBox 函数获

取用户输入的索引号。使用 Sheets(Val(a))语句引用该索引号对应的工作表。在第 05 行代码中使用 Sheets(Val(a)).Visible 语句获得工作表 Visible 属性的值，使用 IF 语句来判断工作表的显示状态。如果该值不为 0，说明工作表处于可见状态，则在第 05 行设置 Visible 属性为 xlHidden 将其隐藏，否则，在第 08 行将 Visible 值设置为-1 使工作表可见。

🔔警告：在本例中，如果没有错误处理语句 On Error GoTo check，则当单击输入对话框的“取消”按钮或输入其他非数字字符串时，由于与程序中引用工作表对象的 Sheets（工作表编号）语句需要的参数类型不符，程序将报错而无法运行。处理的方法就是像范例这样，使用 On Error 语句，在遇到错误输入时让程序跳转到最后退出过程即可。

12.4　复制和移动 Excel 工作表

在创建 Excel 数据表时，常常需要对数据表进行复制和移动操作。在 VBA 中，使用 Copy 方法和 Move 方法，能够方便地实现工作表对象的复制和移动。

12.4.1　复制 Excel 工作表

要在 VBA 程序中对指定的工作表进行复制，可以使用 Worksheet 对象的 Copy 方法，其语法格式如下：

```
对象.Copy(Before,After)
```

参数说明如下所示。

- ❑ Before：在进行工作表复制时，工作表将复制到此参数指定的工作表之前。
- ❑ After：在进行工作表复制时，工作表将复制到此参数指定的工作表之前。

例如，当需要将“Sheet1”工作表复制到“Sheet3”工作表之前，可以使用下面的语句来实现。

```
Worksheets("Sheet1").Copy After:=Worksheets("Sheet3")
```

🔔注意：在进行复制操作时，Before 参数和 After 参数只能使用一个。如果这两个参数都没有使用，则将把工作表复制到新的工作簿中。

【范例 12-7】 使用名为“成绩表.xlsx”素材表格，将工作簿中的 3 个成绩表复制到一个新的工作簿中，完成复制后保存工作簿，代码如下所示。

```
Sub 复制工作表()
   ActiveWorkbook.Sheets.Copy                       '复制工作簿中所有工作表
   ActiveWorkbook.SaveAs "成绩表副本.xlsx"           '保存复制的工作表
 End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行

程序，当前工作簿的所有工作表复制到新工作簿中，同时新工作表保存名为“成绩表副本.xlsx”的文件，如图 12.12 所示。

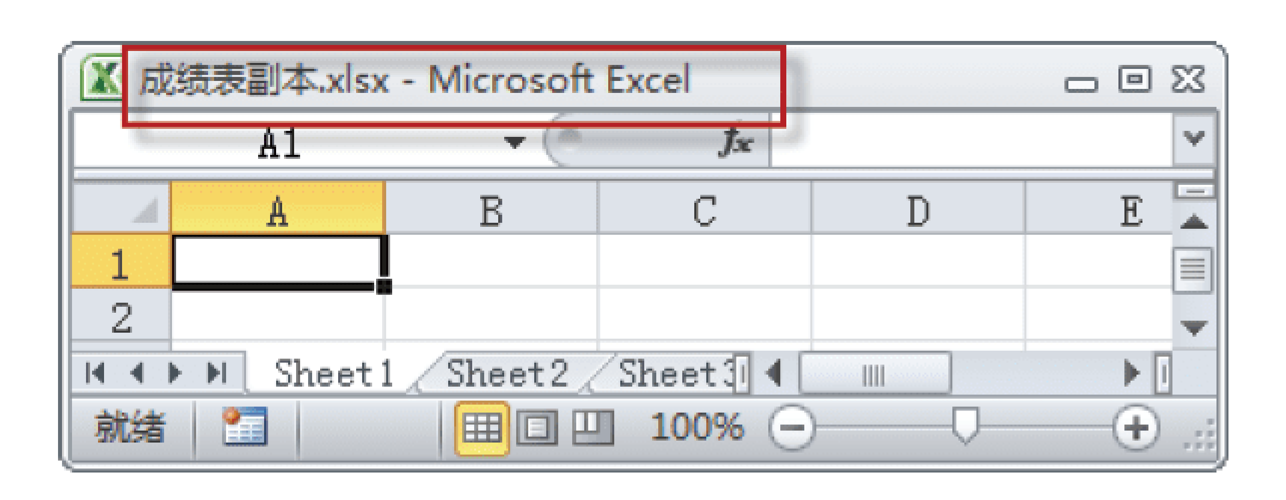


图 12.12　复制工作表到新工作簿中

【代码解析】本范例演示不带参数的 Copy 方法的使用。在程序中，Activeworkbook.sheets 可实现对当前工作簿的所有工作表的引用，使用 Copy 方法将这些工作表复制到新的工作簿中。然后使用 SaveAs 方法保存该工作簿。

提示： 复制工作簿中所有工作表，也可以使用 For…Next 循环遍历所有的工作表来进行复制，但这种方法没有范例采用的方法效率高。

12.4.2　移动 Excel 工作表

移动工作表是指改变工作表在工作簿中的排列位置或将指定工作表移动到新工作簿中。使用 Worksheet 对象的 Move 方法能实现指定工作表的移动，其一般语法结构如下所示。

```
对象.Move (Before,After)
```

Move 方法的参数与 Copy 方法的参数意义相同，与工作表的复制相比，移动工作表相当于在完成工作表的复制后删除当前的工作表。例如，将名为“工资”的工作表复制到名为“Sheet1”的工作表之前，可以使用下面的语句。

```
Sheets("工资").Move Before:=Sheets("Sheet1")
```

与 Copy 方法一样，使用 Move 方法时如果不带参数，则工作表将移到新的工作簿中。例，将一个名为“七年级成绩表”的工作表移到新的工作簿中，可以使用下面语句来实现。

```
Sheets("七年级成绩表").Move
```

【范例 12-8】 使用名为“成绩表.xlsx”素材工作簿，编写程序将名为“成绩表 1”的第一个工作表换名复制到工作簿的最后，使用的工作表名由用户从输入对话框中输入，代码如下所示。

```
Sub 移动工作表示例()
    Dim sht As Worksheet, n As String                    '变量声明
    n = InputBox("请输入新工作表名")                      '输入新工作表名
    For Each sht In Worksheets                           '遍历所有工作表
        If sht.Name = n Then                             '是否有同名工作表存在
```

```
06              MsgBox "工作表已经存在，重命名将无法进行。"  '有，则给出提示
07          End If
08      Next
09      Sheets(1).Copy before:=Sheets(1)                   '将工作表复制到最前面
10      Sheets(1).Name = n                                 '工作表更名
11      Sheets(n).Move after:=Sheets(Sheets.Count)         '移动更名后的工作表
12  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序首先显示输入对话框要求用户输入新工作表名，如图 12.13 所示。完成输入关闭对话框后，第一个工作表被复制到工作簿的最后，并且使用刚才输入的工作表名，如图 12.14 所示。

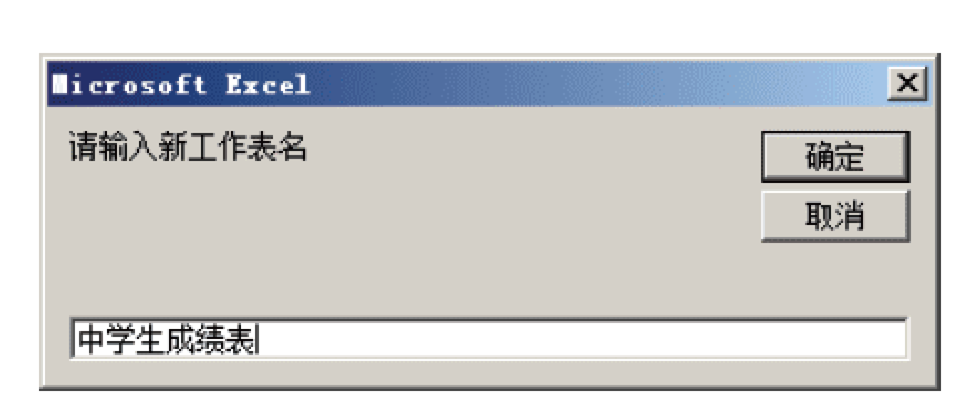


图 12.13　输入工作表名

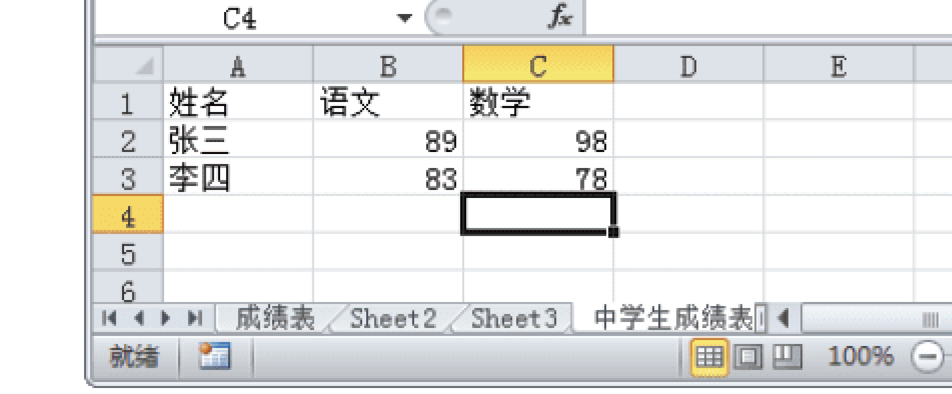


图 12.14　复制新的工作表

【代码解析】本例给出了编程实现工作表换名复制的一种方法，即复制需要换名的工作表，然后将其换名后移动到指定的位置。在本例的程序中，使用 InputBox 函数来获得用户输入的名称，为了避免新工作表名与已有的工作表重名，使用 For Each …In Next 结构遍历所要所有工作表，用 If 语句来判断是否有与输入同名的工作表存在。如果没有，则执行第 09 行以后的代码，实现换名复制。在代码的第 09 行，“成绩表 1”工作表被复制到了工作簿的最前面，然后通过修改 Name 属性值来修改工作表名，最后使用 Move 方法将其移动到工作簿的最后。

12.5　打印 Excel 工作表

在 Excel 中当然可以使用页面设置功能对打印页面进行设置并将工作表打印出来。在 VBA 中，使用 PrintOut 方法可以在程序中实现工作表的打印以及对打印页面进行设置。VBA 的 PrintOut 方法的语法格式如下：

```
表达式.PrintOut(From, To, Copies, Preview, ActivePrinter, PrintToFile,
Collate, PrToFileName, IgnorePrintAreas)
```

参数说明如下所示。

- ❑ From：此参数用于设置打印的开始页号。如果省略此参数，则从起始位置开始打印。
- ❑ To：此参数用于设置打印的终止页号。如果省略此参数，则打印至最后一页。
- ❑ Copy：设置打印份数。如果省略此参数，则只打印一份。
- ❑ Preview：此参数设置为 True 时，Microsoft Excel 将在打印对象之前调用打印预览。

如果为 False（或省略该参数），则立即打印对象。

- ActivePrinter：设置活动打印机的名称。
- PrintToFile：此参数如果设置为 True，则打印到文件。如果没有指定 PrToFileName，Microsoft Excel 将提示用户输入要使用的输出文件的文件名。
- Collate：此参数如果设置为 True，则逐份打印多个副本。
- PrToFileName：如果 PrintToFile 设为 True，则此参数指定要打印到的文件名。
- IgnorePrintAreas：此参数如果设置为 True，则忽略打印区域并打印整个对象。

进行工作表打印时，离不开对页面的设置。设置页面需要使用 PageSetup 对象来实现，该对象包含了对页面设置需要的所有属性，如底部边距、纸张大小以及页眉样式等。如果打印工作表时，需要在每页的右上角打印工作表名，可以使用下面语句实现：

```
Worksheets("Sheet1").PageSetup.RightHeader = "&F"
```

【范例 12-9】 打印“成绩表.xlsx”素材表格的第一个工作表的“A1:F9”区域，打印前设置所有的页边距和页眉，代码如下所示。

```
Sub 移动工作表示例()
    With Worksheets(1).PageSetup
        .LeftMargin = Application.InchesToPoints(1)  '设置左边距为 1 磅
        .RightMargin = Application.InchesToPoints(1)'设置右边距为 1 磅
        .TopMargin = Application.InchesToPoints(1.7)'设置顶端距为 1 磅
        .BottomMargin = Application.InchesToPoints(1)
                                                    '设置底端边距为 0 磅
        .HeaderMargin = Application.InchesToPoints(0.7)
                                                    '设置上边距为 1 磅
        .CenterHeader = "&""隶书,加粗""&20 七年级期中成绩表" '设置页眉文字
        .PrintArea = "$A$1: F$9"                      '设置打印区域
    End With
    Worksheets(1).PrintPreview                        '打印预览
    Worksheets(1).PrintOut Copies:=1                  '打印一份
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行后将显示打印预览，如图 12.15 所示。单击“关闭打印预览”按钮关闭“打印预览”窗口，工作表开始打印。

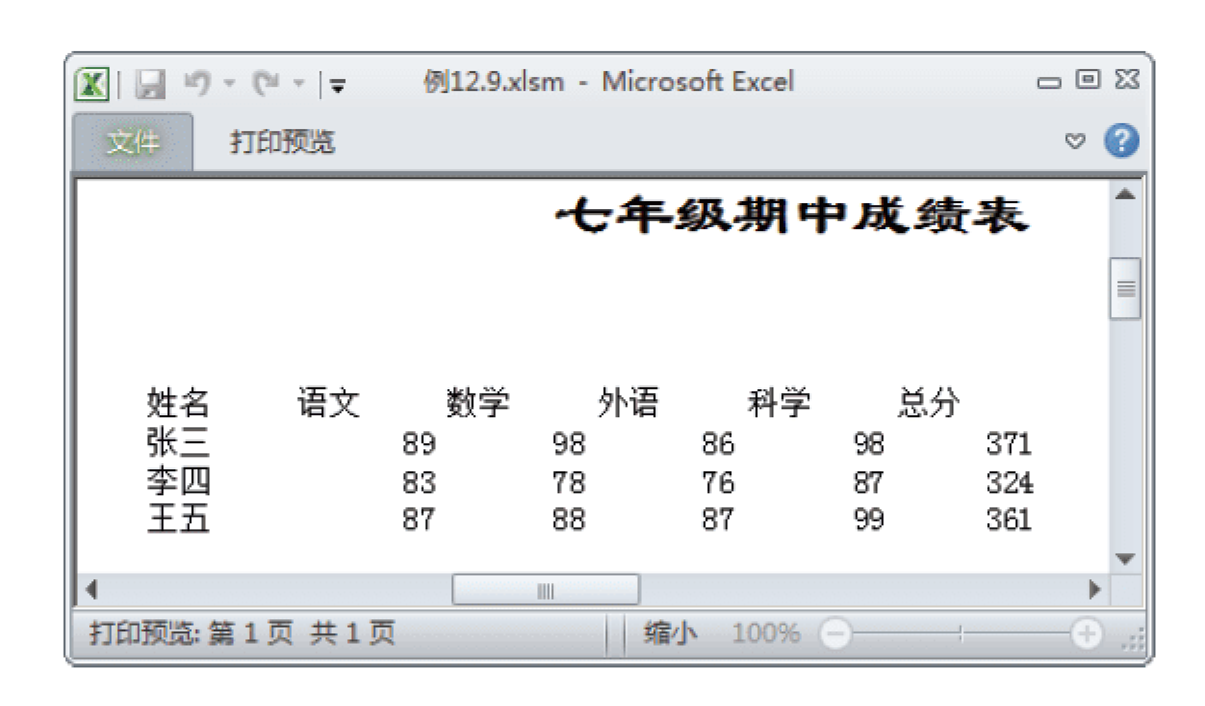


图 12.15　预览打印效果

【代码解析】本示例程序设置打印表格时的左右和上下边距、设置页眉文字样式和打

印区域，同时程序能够预览打印效果。代码从 02~10 行使用 With 结构来设置 PageSetup 对象的属性，修改这些属性的值可以实现对打印页面设置。程序的第 08 行设置页眉居中，并且设置了页眉的字体和文字大小。第 09 行通过对 PrintArea 属性的设置指定了打印的工作表区域。第 11 行使用 PrintPreview 方法来预览打印效果，第 12 行通过设置 Copies 参数指定了打印份数。

警告： 这里的程序要能够运行，系统的 Print Spooler 的进程必须要启用。该进程用于管理所有本地和网络打印队列及控制所有打印工作。如果此进程没有开启，程序运行时将提示无法设置 PageSetup 类的属性，程序将无法运行。当然在 Excel “页面布局”选项卡的大多数命令也都不可用。

12.6　工作表的其他操作

Workbook 对象和 Workbooks 对象集提供了丰富的属性和方法来满足针对工作表的编程需要，本节将介绍编程设置 Excel 滚动区域、编辑工作表批注和删除空白工作表的方法。

12.6.1　设置 Excel 工作表的滚动区域

在 Excel 中，当工作表中数据较多，窗口无法完全显示时，可以通过拖动滚动条来显示那些没有完全显示的数据。在 VBA 中，可以通过对 Worksheet 对象的 ScrollArea 的属性进行设置来指定拖动滚动条允许显示的单元格区域。同时，设置滚动区域对工作表也有保护作用，非指定区域内的其他单元格将无法进行编辑。

【范例 12-10】 将“成绩表.xlsx”素材表格的“A1:G11”区域设置为滚动区域，代码如代码示例 12-10 所示。

```
Sub 设置滚动区域()
    Sheet1.ScrollArea = "A2:G11"                  '设置滚动区域
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行后，拖动工作簿窗口中的滚动条，超过指定区域的单元格将无法显示。同时，只有指定区域的单元格才能被选择，并进行编辑，如图 12.16 所示。

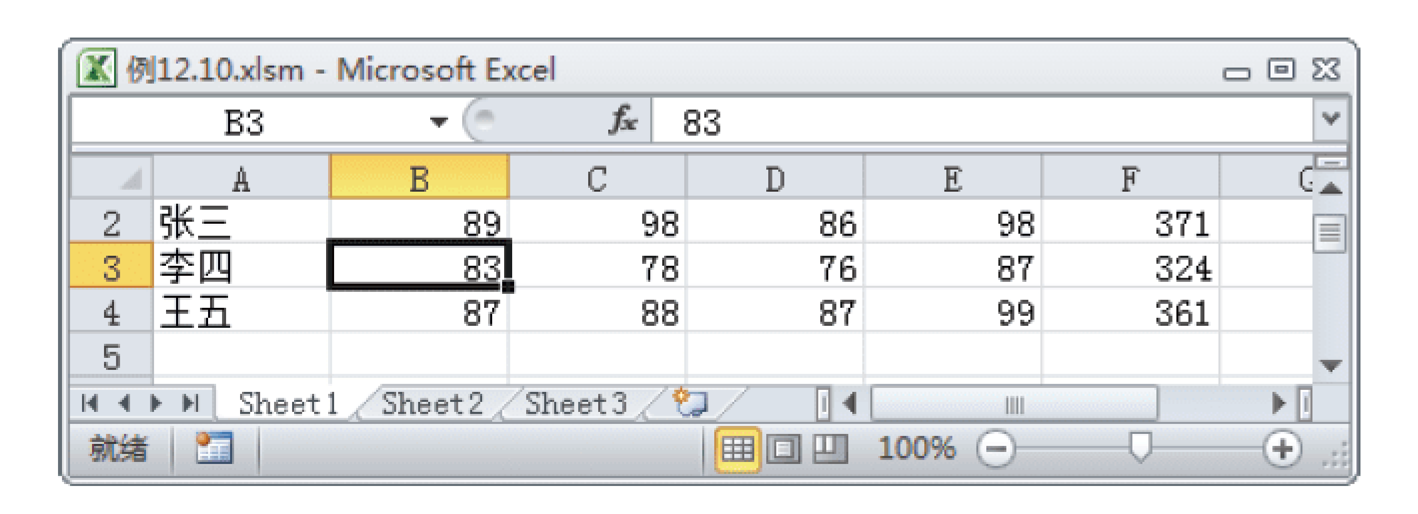


图 12.16　指定区域的单元格能够显示

【代码解析】在代码中使用 ScrollArea 属性来设置工作表的滚动区域，该属性能够将指定的单元格区域引用设置为允许或禁止滚动的区域，设置后用户不能选定滚动区域之外的单元格。

提示：如果要取消对滚动的限制，只需要将 ScrollArea 属性设置为空字符串即可。

12.6.2　查看 Excel 工作表中的批注

使用 Excel 的“审阅”选项卡的“批注”组，可以实现对单元格批注的添加、删除和隐藏等操作。VBA 提供了一个 Comment 对象，使用该对象的属性和方法可以对工作表中的批注进行管理。如读取 Comment 对象的 Author 属性可以获得批注作者的信息，设置 Visible 属性值可以控制批注是否可见，使用 Delete 方法可以删除批注。下面通过一个实例来介绍使用 Comment 对象的属性来查看工作表中批注信息的方法。

【范例 12-11】 程序运行后，用户能够通过向输入对话框中输入数字来选择需要查看的标注，代码如下所示。

```
01   Sub 批注管理()
02       n = InputBox("当前工作表中有" & Sheet1.Comments.Count &
03       "条批注，你需要查看哪一条？")              '输入需要查看批注的编号
04       If n > 0 And n <= Sheet1.Comments.Count Then
                                                  '如果输入的是小于批注总数的数字
05           MsgBox "第" & n & "条批注:" & Chr(13) _
06           & "作者: " & Sheet1.Comments(1).Author & Chr(13) _
07           & "批注内容: " & Sheet1.Comments(1).Text      '显示标注信息
08       End If
09   End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行后将首先显示输入对话框显示批注的提示，并要求用户输入需要查看批注的编号，如图 12.17 所示。输入编号后，单击“确定”按钮关闭对话框，将显示批注的内容，如图 12.18 所示。

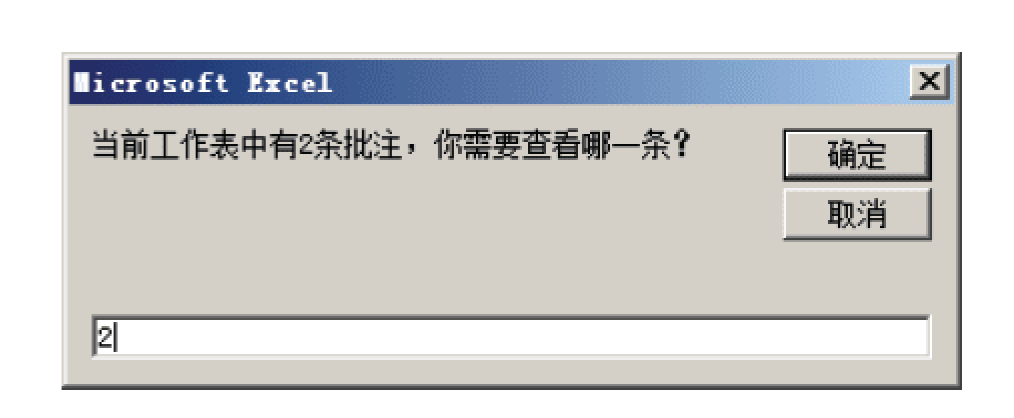



图 12.17　输入对话框输入批注的编号



图 12.18　显示批注内容

【代码解析】本范例用于用户查阅当前工作表中的批注。程序使用 InputBox 函数来实现用户对需要查看批注的选择，使用 MsgBox 函数来显示批注的内容。在第 02 行代码中，使用 Count 方法来获得当前工作表中批注的总数，并在输入对话框中显示这个总数。在第 04 行程序对用户输入进行判断，保证只有输入数字在 0 至批注总数之间时，才能显示批注

内容。第 06 行代码中，使用 Author 属性值获得批注作者。第 07 行中使用 Text 方法获得批注的内容。

提示： 在 VBA 中，每一个批注都是一个 Comment 对象。在工作表中，Comment 对象组成一个 Comments 集合，因此本例的操作就是在对这个 Comments 集合的操作，如 Comments（1）语句指的就是对象集中的第一个对象。如果在工作表中没有批注，则这个集合是空的。

12.6.3　删除空白 Excel 工作表

在处理数据表时，有时会在工作簿中留下一些空白的工作表，即所有单元格均为空的工作表。在完成数据处理后，这些不会被使用的空白工作表需要删除。下面介绍通过 VBA 编程来快速删除这些工作表的方法。

【范例 12-12】 在工作簿中创建新表格并输入数据，在新表格创建完成后，删除 Excel 的 3 个默认的空白的工作表（即工作表“Sheet1”、“Sheet2”和“Sheet3”），代码如下所示。

```
01  Sub 删除空白表()
02      Dim s As Worksheet                                   '声明数组变量
03      Application.DisplayAlerts = False                    '关闭删除警告
04      For Each sht In ThisWorkbook.Sheets                  '遍历所有工作表
05          If WorksheetFunction.CountA(sht.Cells) = 0 Then sht.Delete
                                                             '删除空白工作表
06      Next sht
07      Application.DisplayAlerts = True                     '重新打开删除警告
08  End Sub
```

【运行结果】启动 Excel，看到空白工作簿中插入 3 个工作表，并将其命名为“成绩表 1”、“成绩表 2”和“成绩表 3”。在这些成绩表中输入学生成绩，Excel 默认的 3 个工作表为空白工作表，如图 12.19 所示。切换到 Visual Basic 编辑器，在工程资源管理器中创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序将删除空白的工作表，如图 12.20 所示。

【代码解析】本例可用于完成数据录入和处理后的工作簿的清理。在程序中，使用 For Each…In Next 循环来遍历工作簿中的所有工作表，如果工作表中非空白单元格数为 0，则判定为空白工作表，将其删除。在第 05 行，使用 WorksheetFunction 对象的 CountA 方法来统计非空单元格的数量，其后的参数 sht.Cells 表示工作表中所有单元格。以 CountA 获得的值作为判定是否为空白表的条件，如果其值为 0，则该工作表为空白工作表，用 Delete 方法删除。

警告： 程序的第 03 行代码关闭了 Excel 的提示功能。如果没有这个语句，那么每次执行第 05 行删除空白表时，Excel 都会提示是否真的删除。而在完成工作表的删除后，应该恢复被关闭的警告功能，因此在第 07 行将 DisplayAlerts 设置为 True。

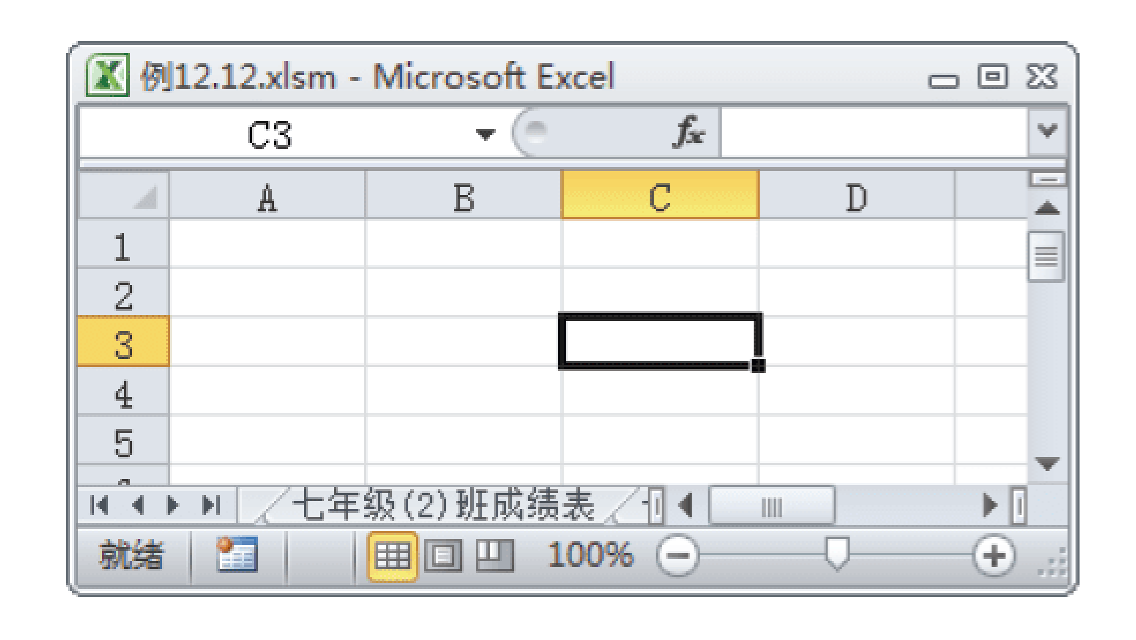


图 12.19　创建成绩表

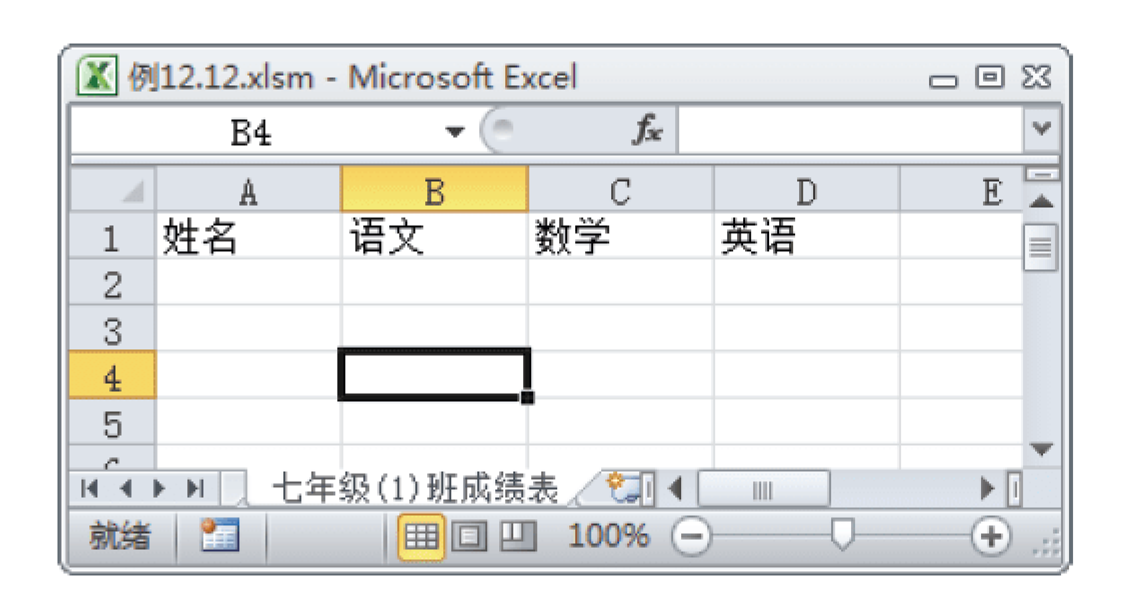


图 12.20　空白工作表被删除

12.7　使用 Excel 工作表事件

在 Excel 中，要控制用户在工作表中的操作，需要为工作表事件添加代码。工作表事件的编程，在开发 Excel VBA 应用程序时经常用到。与工作簿事件相比，工作表事件要少一些，只有 9 个。工作表事件是在工作表被激活、用户对工作表进行修改以及更改工作表上的单元格和数据透视表时被触发。本节将对工作表常用事件及其使用进行介绍。

12.7.1　使用 Excel 工作表激活事件

Activate 事件是工作表被激活时发生的事件，将程序放置于事件中，能够实现工作表激活时程序的自动运行。

【范例 12-13】 激活工作表时工作表中指定单元格区域中的数据自动排序，代码如下所示。

```
Private Sub Worksheet_Activate()
    Range("A2:G31").Sort Key1:=Range("G1"), Order1:=xlDescending
                                                         '按降序排序
End Sub
```

【运行结果】在 Visual Basic 编辑器中为第一个工作表添加上面的代码，切换到 Excel 窗口，激活此工作表，工作表中的数据将自动按“总分”的降序排列，如图 12.21 所示。

	A	B	C	D	E	F	G
1	姓名	班级	语文	数学	外语	科学	总分
2	张三	1	89	98	86	98	371
3	王五	3	87	88	87	99	361
4	李四	2	83	78	76	87	324
5	赵六	1	87	67	68	98	320

图 12.21　工作表中的数据按总分的降序排列

【代码解析】本范例使用工作表的 Activate 事件来实现激活工作表后数据自动排序。在程序中，使用单元格对象的 Sort 方法来实现排序。本程序使用了 Sort 方法的两个参数，一

个是 key1 参数，其用于指明排序字段，本程序用的是“总分”字段。第二个参数是 Order，其设置排序方式。本程序将 Order 参数设置为 xlDescending，即降序。

🔔提示：如果需要数据按照升序排列，可将 Order 参数设置为 xlAsecending。

12.7.2　使用 Excel 单元格更改事件

当用户更改工作表中的单元格或外部链接引起单元格的变化时，会触发工作表的 Change 事件。Change 事件带有一个 Target 参数，在事件发生时，当前操作的单元格 Range 对象将会传递给这个参数。使用这个参数将能了解工作表中哪些单元格发生了改变。

【范例 12-14】 使用 Change 事件程序检验单元格输入数据的合理性，代码如下所示。

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 6 Then                   '判定改变的是否是第 6 列单元格
        If Target.Value > 150 Or Target.Value < 0 Then
                                                '判定分数是否超过了 150 或低于 0
            Target.Select                       '选择此单元格
            MsgBox "科学分数输入错误！"            '提示分数输入错误
            Target = ""                         '单元格设为空
        End If
    End If
    If Target.Column > 1 And Target.Column < 6 Then
                                                '判定是否是第 2 列到第 5 列单元格
        If Target.Value > 120 Or Target.Value < 0 Then
                                                '判定分数是否超过 120 或低于 0
            MsgBox "分数输入错误！"               '给出提示
            Target = ""                         '单元格清空
            Target.Select                       '选择单元格
        End If
    End If
    If Target.Column = 1 Then
        MsgBox "请不要更改此行数据"               '是第一列给出提示
    End If
End Sub
```

【运行结果】 在 Visual Basic 编辑器中为第一个工作表添加上面的代码，切换到 Excel 窗口，修改工作表中单元格数据。在各科的分数中输入或修改分数时，如果分数超过了满分，则程序会给出提示，如图 12.22 所示。同时将该单元格清空，并选择该单元格。如果修改了第一列的数据，程序同样给出提示对话框提示数据不能修改。

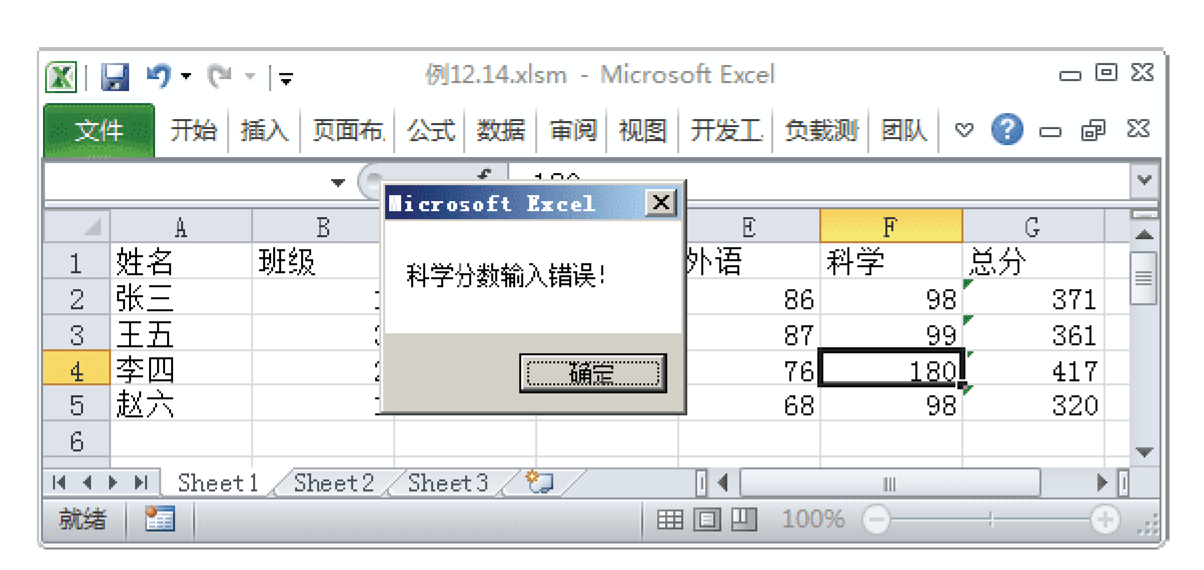


图 12.22　输入分数错误时的提示

【代码解析】本范例通过 Change 事件代码来实现对工作表中单元格数据合法性的检验。当工作表中单元格数据发生变化时，事件将触发。此时，根据 Target 参数来确定更改的是哪列单元格的数据，根据不同列学科的总分来判定输入是否合理，分别给出提示。

在程序中，事件的 Target 参数将返回一个 Range 对象，通过对象的 Column 属性得到被修改单元格的列号，使用 Target.Value 语句获得单元格中的数据，使用 Target.Select 语句来实现对单元格的选取。

本程序通过 If 结构的嵌套来显示对输入分数的判断。成绩表中科学分数的满分是 150 分。其他各科的满分是 120 分。在第 02 行至 08 行的程序段中，首先判断当前改变的是否是科学成绩，然后判断是否在允许的分数范围之外，如果是则给出提示并清除错误分数。第 09~15 程序段对其他学科成绩的正确性进行检验，编程方法与前面科学成绩的检验相同。

注意：在工作簿中往往有多个工作表，工作表事件代码必须写在对应的工作表对象中。如本例中对“Sheet1”对象的事件编程，代码必须写在该工作表对象的“代码”窗口中。

12.7.3　使用 Excel 工作表的选择区域变化事件

SelectionChange 事件是在对工作表的单元格的选择发生改变后产生的事件。使用该事件可以方便地实现对选择的单元格或单元格区域进行自动操作。与 Change 事件一样，SelectionChange 事件带有参数，其参数是一个 Range 变量，表示被选择的单元格区域。

【范例 12-15】 使用 SelectionChange 事件实现高亮显示选择的单元格所在的行。要求选择时，整行文字格式也随之发生改变，代码如下所示。

```
01  Private Sub Worksheet_SelectionChange(ByVal Target As Range)
02      Dim r As String                                  '声明变量
03      Cells.FormatConditions.Delete                    '删除当前条件格式
04      With Target.EntireRow                            '对单元格所在的行进行操作
05          r = .Address                                 '获取单元格地址
06          .FormatConditions.Delete                     '删除单元格所在行的格式
07          .FormatConditions.Add Type:=xlExpression, _
08          .Formula1:="=COUNTA(" & r & ")>0"            '添加条件格式
09          .FormatConditions(1).Font.Bold = True        '设置行文字为粗体
10          .FormatConditions(1).Font.Italic = True'设置行文字倾斜显示
11              .FormatConditions(1).Interior.ColorIndex = 20
                                                         '设置行填充颜色
12      End With
13  End Sub
```

【运行结果】在 Visual Basic 编辑器中为第一个工作表添加上面的代码，切换到 Excel 窗口，在工作表中选择单元格，单元格所在的行文字样式发生改变，同时整行会被色条框选。本实例程序运行效果如图 12.23 所示。

【代码解析】本实例实现当选择某个单元格时，单元格所在的行会被填充颜色，并且文字样式发生改变。在本例中，使用了 FormatConditions 对象集的属性和方法，该对象集代表一个区域内所有条件格式的集合。在第 03 行中，使用该对象集的 Delete 方法删除所

有单元格的条件格式，然后在第 04~13 行使用 With…End With 结构重新设置选择单元格所在行的条件格式，从而实现程序的功能。

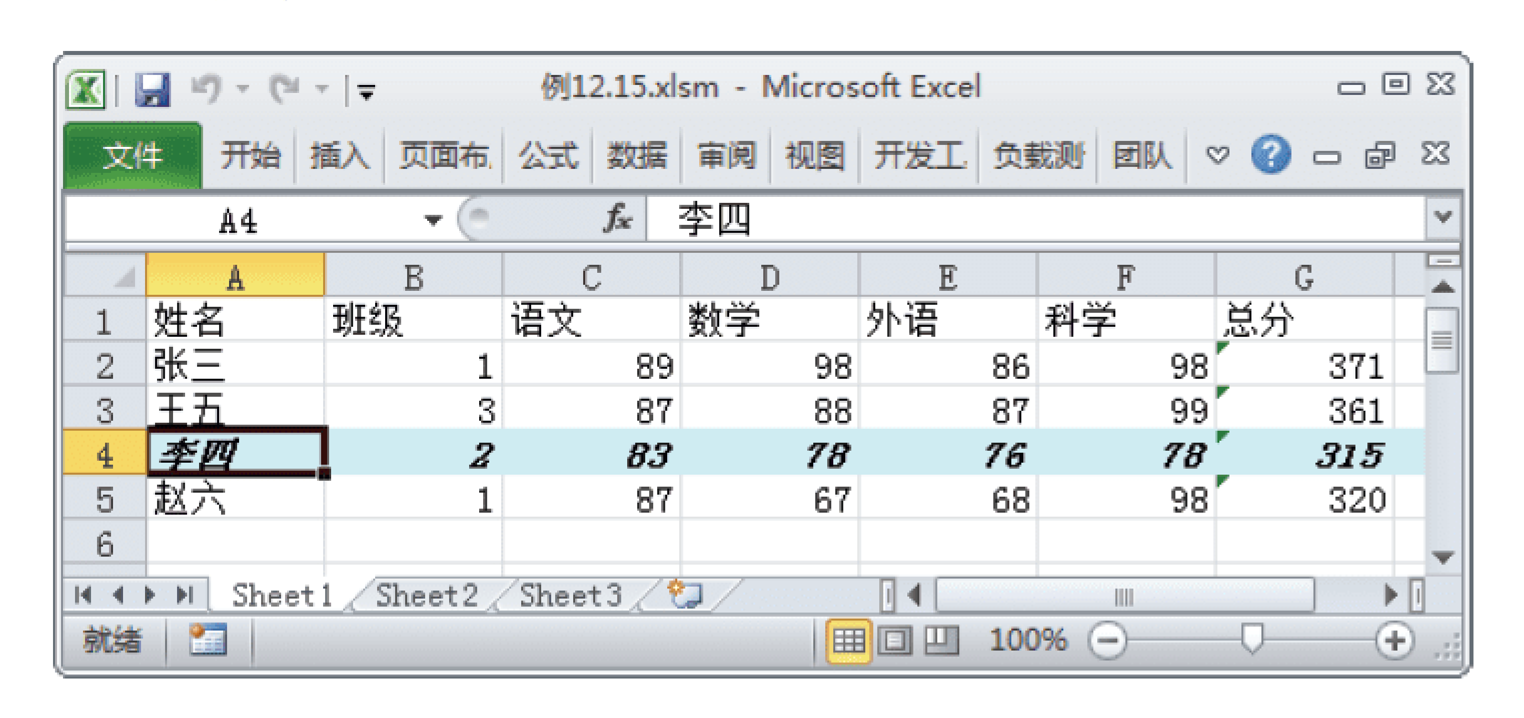


图 12.23　程序运行效果

代码第 04 行中的 Target.EntireRow 表示选择单元格所在的整行。第 05 行的 Address 属性代表对区域的引用。在第 07 行使用 Add 方法来添加一个新的条件格式，Type 参数指明格式是基于表达式，而 Formula1 参数指定了与条件格式相关联的条件表达式，指定了条件格式生效的条件是当前行不是空行。在第 09~11 行设置单元格的样式。

警告： 在本实例代码中第 03 行代码是必须的，如果没有这行代码，更换单元格选择后，上一行的样式将仍然存在。

12.7.4　使用 Excel 工作表右击事件

在工作表上右击将触发工作表的 BeforeRightClick 事件，此事件的发生将优于 Excel 默认的鼠标右键操作。利用这一特点可使用自己编写的事件代码来取代默认的右键关联菜单的操作，或者对右键菜单进行修改。

【范例 12-16】 录制一个宏，该宏能够设置单元格文字的样式，该宏命名为“FontStyle”。为工作表右键关联菜单添加一个名为“更改单元格文字样式”的命令，使用该命令能够调用“FontStyle”宏来设置文字样式，代码如下所示。

```
01  Private Sub Worksheet_BeforeRightClick(ByVal Target As Range, Cancel As
    Boolean)
02      For Each n In Application.CommandBars("cell").Controls
                                                    '遍历关联菜单中所有项
03          If n.Tag = "mycommand" Then n.Delete    '判断如果存在命令项则删除
04      Next
05      With Application.CommandBars("cell").Controls.Add _
06      (Type:=msoControlButton, Before:=1, Temporary:=True)
                                                    '添加一个新的菜单项
07          .Caption = "更改单元格文字样式"         '设置菜单命令名
08          .OnAction = "FontStyle"                 '指定执行宏
09          .Tag = "mycommand"                      '设置命令标签
10      End With
11  End Sub
```

【运行结果】在 Visual Basic 编辑器中为第一个工作表添加上面的事件代码，切换到

Excel 窗口，在工作表中右击，在弹出的快捷菜单中会出现“更改单主格文字样式”命令，选择该命令可将单元格文字改为指定的样式。本实例程序运行效果如图 12.24 所示。

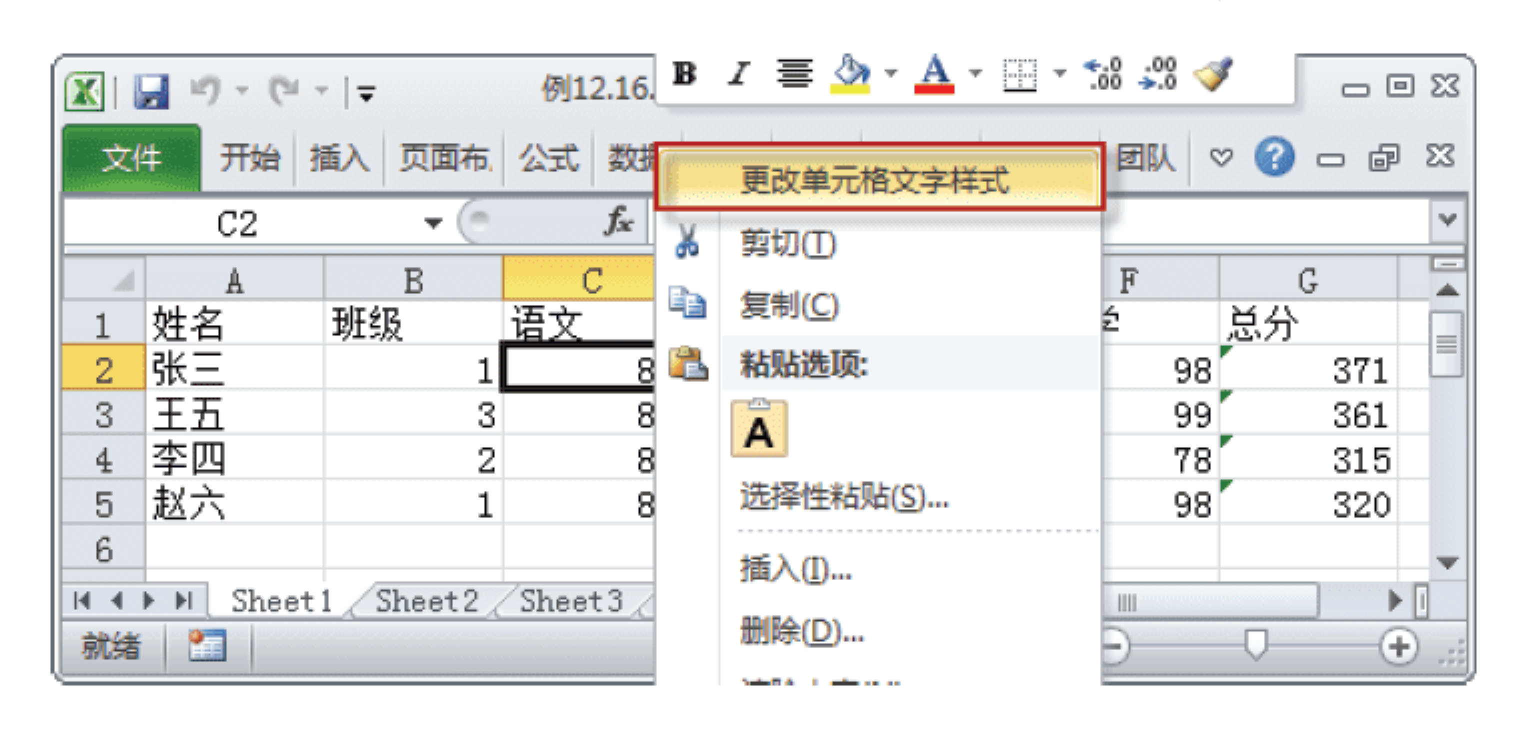


图 12.24　程序运行效果

【代码解析】本实例使用 BeforeRightClick 事件来添加工作表右键关联菜单命令。第 02～04 行代码通过循环结构来遍历所有菜单项，将已经存在的标记为"mycommand"菜单命令删除。第 05～10 行使用 With…End With 结构在右键菜单中添加一个菜单命令，设置菜单命令的名称，指定相关联的宏，设置标记名称。

程序使用了 CommandBars 对象集合，其代表 Excel 应用程序命令栏对象的集合。在第 02 行代码中，CommandBars("cell").Controls 语句将获得工作表命令栏上的所有命令项的集合，然后逐一进行判断，看是否是标记为"mycommand"的项目并将其删除。在第 06 行，使用 Add 方法向菜单对象中添加一个新的控制项。这里使用了 2 个参数，Type 参数用于指定添加对象类型，这里是一个控制按钮，对于菜单来说就是一个菜单项；Before 参数用于指明添加该项的位置，也就是在第 6 项之前。Temporary 参数将该控制项设置为临时控件，即在关闭应用程序时将被删除。

警告：在程序开始时，必须要使用 Delete 方法删除已添加的菜单命令，否则每次右击后，关联菜单中都会添加一个相同的菜单项。

12.8　小　　结

本章介绍了 Excel VBA 中的工作表对象和工作表对象集的使用方法，通过一些典型的实例，介绍了工作表引用的方法、创建和删除工作表的方法、选取和隐藏工作表的方法、复制和移动工作表的方法、打印工作表的方法以及工作表事件的使用和其他一些常见的工作表编程应用。通过本章的学习，读者将掌握常用的工作表编程技巧，能够利用 VBA 解决实际的数据处理问题。

对工作表的编程，经常需要遍历工作簿中的所有对象，使用对象变量是一个提高效率的好方法。同时，通过本章列举的各个工作表编程实例，读者将能体会到 Excel VBA 的程序设计实际上就是使用对象方法和属性来编写代码的过程。Excel 构成元素都有对象与之对应，要掌握这些对象的属性和方法，VBA 的帮助文档是最权威的工具。

12.9　本 章 习 题

1．下面对工作表的引用错误的是哪一个？（　　）

A．sheet1　　B．sheets(1)　　C．Worksheets(1)　　D．sheets(成绩表 1)

2．选择下面程序运行结果。（　　）

```
Sub test()
    Dim p As String
    Sheets(Array(1, 3)).Select
    For Each r In Workbooks(1).Windows(1).SelectedSheets
        r.Delete
    Next
End Sub
```

A．工作簿中所有工作表被删除　　B．工作簿的前 3 个工作表被删除

C．工作簿的第一个和第三个工作表被删除　　D．工作簿的第一个工作表被删除

3．选择下面程序的运行结果。（　　）

```
Sub test()
    Sheets(1).Copy before:=Sheets(1)
    Sheets(1).Name = "成绩表 1"
    Sheets(n).Move after:=Sheets(Sheets.Count)
End Sub
```

A．在工作簿的最后获得一个名为“成绩表 1”的工作表

B．在工作簿最前面获得一个名为“成绩表 1”的工作表

C．将原名为“Sheet1”的工作表更名为“成绩表 1”

D．将原名为“Sheet1”的工作表移动到工作簿的最后

4．选择下面程序运行结果？（　　）

```
Private Sub Worksheet Activate()
    Dim sht  As Worksheet
    For Each sht In Sheets
        sht.Protect 123
    Next
End Sub
```

A．名为“123”的工作表添加保护

B．每一个工作表添加保护，密码为“123”

C．第一个工作表添加保护，密码为“123”

D．对名为“sht”工作表添加保护，密码为“123”

5．编写程序，使激活的工作表标签以绿色显示以示强调。

【提示】当前工作簿有 3 个工作表，使用工作表的 Activate 事件来激活程序，3 个工作表都需要添加 Activate 事件响应程序。程序的结构基本相同，那就是激活一个工作表改变标签颜色，同时将另外 2 个工作表的标签颜色恢复正常。

6．在某一列输入数据时，报告出现的重复数据的地址。

【提示】在输入数据后能够检测该列中是否有相同的数据，这个可以通过编写工作表对象的 Change 事件程序来实现。在本练习中，输入的数据在工作表的第一列，程序首先需要判断是否在第一列输入，如果是第一列，然后判断输入的数据与此列数据是否有重复。当存在重复时，应该遍历整列找到重复数据的地址，将其显示出来。这里，判断是否具有相同的数据，可以看成是当前输入相同的数据有几个，如果超过 1 个，则说明有相同的数据存在。要完成这样的工作，可以使用 WorksheetsFunction 对象的条件计数方法（即 CountIf）来实现。

第 13 章　单元格对象

单元格对象是 Excel 中最常用的对象之一。在 Excel 对象模型中 Range 对象表示单元格对象，通过 Range 对象可以引用 Excel 文档中的单元格、删除单元格、插入单元格内容、清除单元格中内容、保护单元格、查找单元格中的数据，以及设置单元格中的格式和筛选单元格数据。本章的主要内容和学习目的如下：

- 掌握使用 Range 对象引用 Excel 单元格；
- 掌握通过 Range 对象引用 Excel 单元格的添加、删除和复制等操作；
- 掌握通过 Range 对象在 Excel 工作表中查找与筛选数据的方法；
- 掌握通过 Range 对象设置 Excel 单元格的格式。

13.1　引用 Excel 单元格

单元格是 Excel 工作表中数据存储的基本单元，引用单元格就是标示工作表中的单元格和单元格区域，以指明单元格在工作表的位置，从而能够对这些单元格中的数据进行操作。在 VBA 中，引用单元格的方式很多，下面对常用的引用方式进行介绍。

13.1.1　引用 Excel 单元格

在 VBA 程序设计时，引用单元格的方式很多，开发者可以使用 Range 属性、使用 Cells 属性、使用记号标示以及使用 RC 样式标示来引用单元格。

在 VBA 中，Worksheet 对象和 Range 对象都有 Range 属性，该属性能够返回一个 Range 对象，使用该属性即可实现对单元格或单元格区域的引用。例如，引用工作表“Sheet1”中的 A3 单元格，可以使用下面的语句：

```
Worksheet("Sheet1").Range("A3")
```

在程序中，可以使用 Cells 属性来引用单元格，这是一种使用行号和列号来引用单元格的方法。例如，引用当前工作表中的 D4 单元格，可以使用下面的语句来实现：

```
Cell(4,4)
Cells(4,"D")
```

注意：这里注意 Cells 与 Range 属性引用单元格的区别，使用 Range 引用单元格，列号在前行号在后。而使用 Cells 时，则是行号写在前，列号写在后。Cells（4，"D"）与 Range（"D4"）指的是相同的单元格。

使用 Cells 引用单元格是一个十分方便的方法。在程序中使用 Cells 属性能够返回一个单元格集合，如果加上参数就是特指某些单元格，如果没有参数，则表示工作表中所有的单元格。例如，下面语句将能够引用“Sheets1”工作表中的所有单元格：

```
Worksheet("sheet1").Cells
```

在 Excel 中，可以使用单元格或单元格区域的名称来实现单元格的引用，具体的操作步骤如下所示：

（1）选择单元格或单元格区域后右击，在弹出的快捷菜单中选择“命名单元格区域”命令打开“新建名称”对话框。在对话框的“名称”文本框中输入区域名称，如图 13.1 所示。

（2）关闭“新建名称”对话框后，在程序中即可使用单元格名称来引用单元格或单元格区域，代码如下所示。

```
Worksheets("Sheet1").Range("myRange")
```

或

```
[myRange]
```

提示：使用单元格名称这种快捷记号来引用单元格区域是十分方便的，特别是像上面示例的第 2 种方式。但使用这种方式只能引用固定值，而无法使用变量。

【范例 13-1】 在工作表的第一行自动填入 2003 年～2008 年年份，在工作表第一列填入 1～12 月，代码如下所示。

```
Sub SetUpTable()
    Worksheets("Sheet1").Activate              '激活工作表
    For y = 3 To 8                             '开始循环
        a = 2000 + y                           '计算年份
        Cells(1, y - 1).Value = a & "年"       '第一行单元格填充数据
    Next
    For m = 1 To 12                            '开始循环
        Cells(m + 1, 1).Value = m & "月"       '第一列单元格填充数据
    Next
End Sub
```

【运行结果】在 Excel 中将工作表“总分”所在的列命名为“总分”，创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行后的效果，如图 13.2 所示。

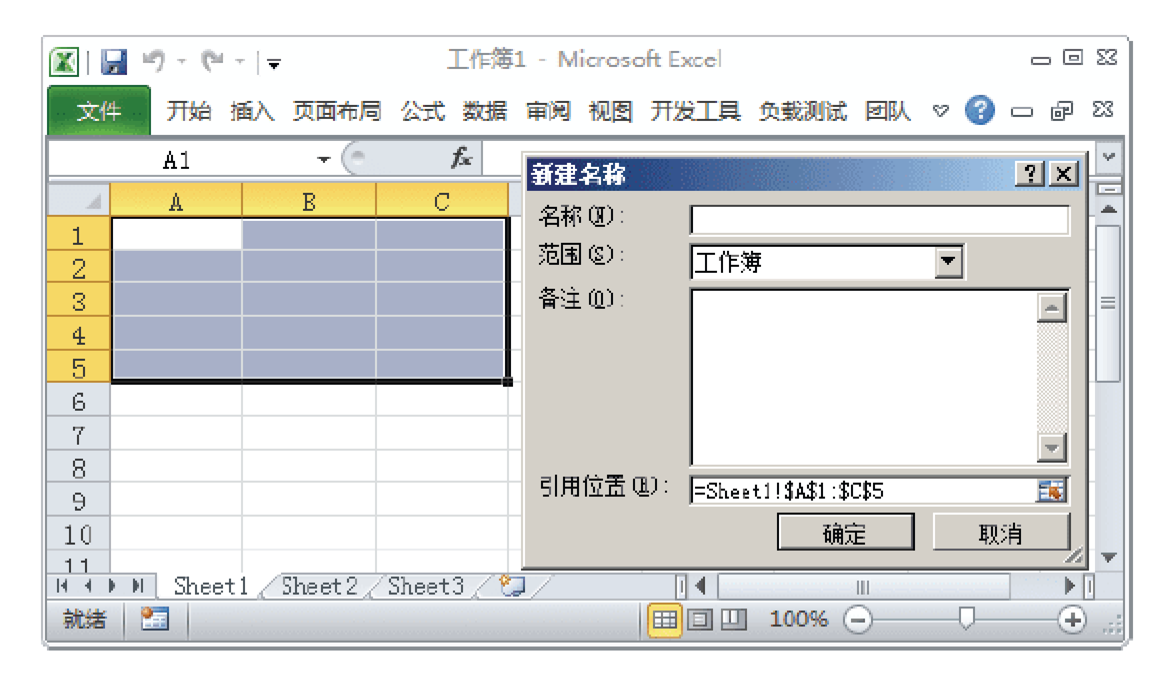


图 13.1　命名单元格区域

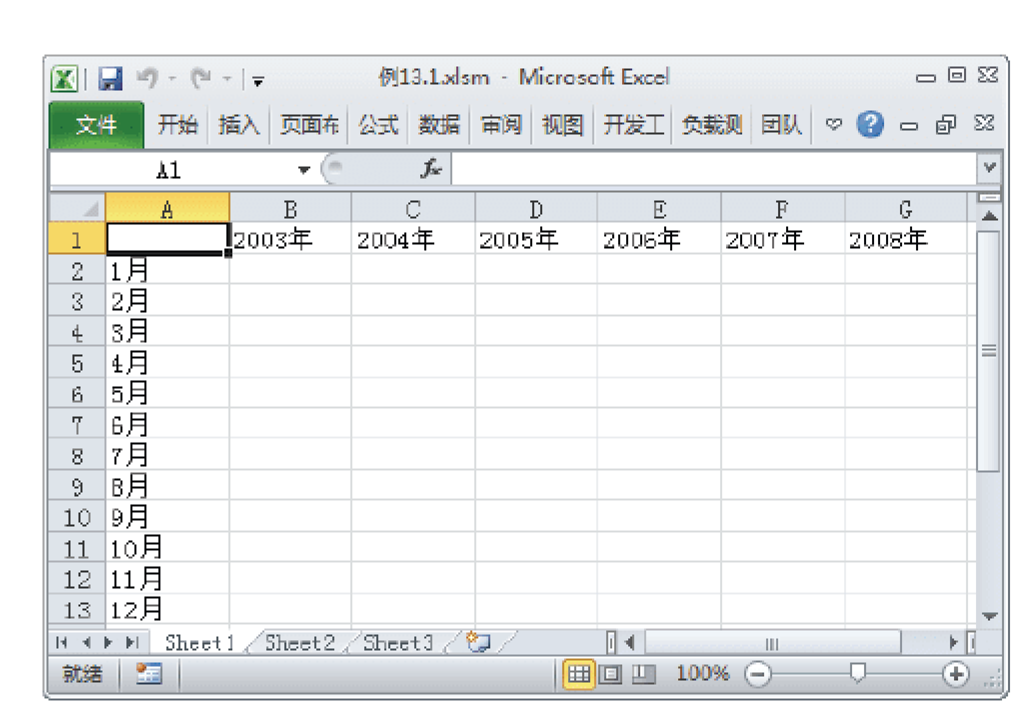


图 13.2　程序运行效果

【代码解析】本范例演示向指定的单元格输入数据的方法。在程序中，使用了两个 For…Next 循环结构来实现对指定的行和列中单元格的引用。在第 05 行和第 08 行引用单元格时，都通过变量来指定行或列。

提示：在编程时，使用工作表名使程序通俗易懂。当在编程中需要使用多个工作表时，可以使用变量来替代索引号，并且能够通过循环结构来处理多张工作表，这样能够使程序更为简捷。

13.1.2　引用 Excel 单元格区域

在 Excel 中，往往需要引用单元格区域。所谓的单元格区域指的是由行或列组成的矩形空间。区域可以是由若干行或列组成，最小的单元格区域当然就是单个的单元格了。

VBA 中要实现单元格区域的引用有多种方法，例如，可以直接使用 Range 引用的单元格区域。当需要引用工作表 A1 至 G15 这个区域时，可以使用下面这些语句来实现。

```
Range("A1:G15")
```

或

```
Range("A1","G15")
```

或

```
Range(Range("A1"):Range("G15"))
```

或

```
Range(Cells(1,1),Cells(7,15))
```

如果需要引用整行，可以使用 Rows 属性来实现，如下面这些示例语句。

```
Rows(4)             '引用单元格的第 4 行
Rows("3:5")         '引用单元格的第 3~5 行
Rows                '引用工作表上所有行
```

当需要应用工作表中的整列时，可以使用 Columns 属性来实现，如下面这些示例语句。

```
Columns(2)          '引用第 2 列
Columns("B")        '引用第 2 列
Columns("A:G")      '引用第 1 列~7 列
Columns             '引用工作表中所有列
```

Range 对象的 EntireColumn 属性和 EntireRow 属性能够返回一个包含指定区域的整行和整列的 Range 对象，使用它们能够实现对整行和整列的引用。例如下面语句将引用当前活动单元格所在的列：

```
ActiveCell.EntireColumn
```

对于工作表中不连续的行或列的引用，可以采用如下面示例语句的方法来实现。

```
Range("1:3,5:8")        '引用第 1~3 列和第 5~8 列
Range("A:C,E:G")        '引用第 A 列到第 C 列和第 E 列到第 G 列
```

使用 Worksheet 对象的 UsedRange 属性表示工作表中已使用的区域，使用该属性可以实现对工作表中所有已使用单元格的引用。例如，使用下面语句将引用 Sheet1 工作表中所有已使用的区域：

```
Sheet1.UsedRange
```

使用 Range 对象的 CurrentRegion 属性可以返回当前的活动单元格区域。所谓当前单元格区域指的是活动单元格所在的矩形区域，这个区域的每一行或每一列至少包含一个非空单元格，每一行和每一列的周围都是空行和空列，如图 13.3 所示。

例如可以使用下面的语句来引用当前的单元格区域：

```
ActiceCell.CurrentRegion
```

在工作表中引用不连续的区域一般有两种方法，一种是使用 Range 直接引用。一种是使用 Application 对象的 Union 方法将区域合并后进行引用。例如，引用 A1 至 C3 单元格区域、D2 单元格和 H3 至 L8 单元格区域，可以使用下面 2 种语句：

```
Range("A1:C3",D2,H3:L8")
```

或

```
Application.Union(Range("A1:C3"),Range("D2"),Range("H3:L8"))
```

使用 Range 对象的 Resize 属性能够将从指定区域处获得一个新的扩展区域，例如，当前选择的单元格为 B2，将单元格区域扩展 5 行 5 列后，选择这个区域，如图 13.4 所示。此时使用的程序代码如下所示。

```
Selection.Resize(5, 5).Select
```

此时，引用这个单元格区域可以使用下面两种方式来实现：

```
Range("D3").Resize(5,5)
```

或

```
ActiveCell.Resize(5,5)
```

注意：如果是扩展一个单元格区域，使用 Resize 属性只是对单元格区域左上角的单元格区域进行扩展。

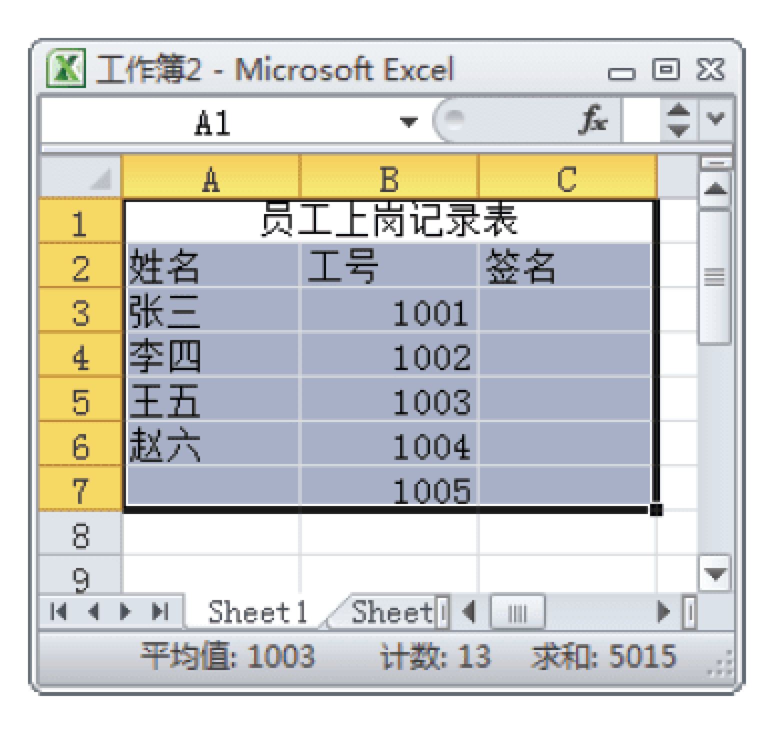


图 13.3　引用当前单元格区域

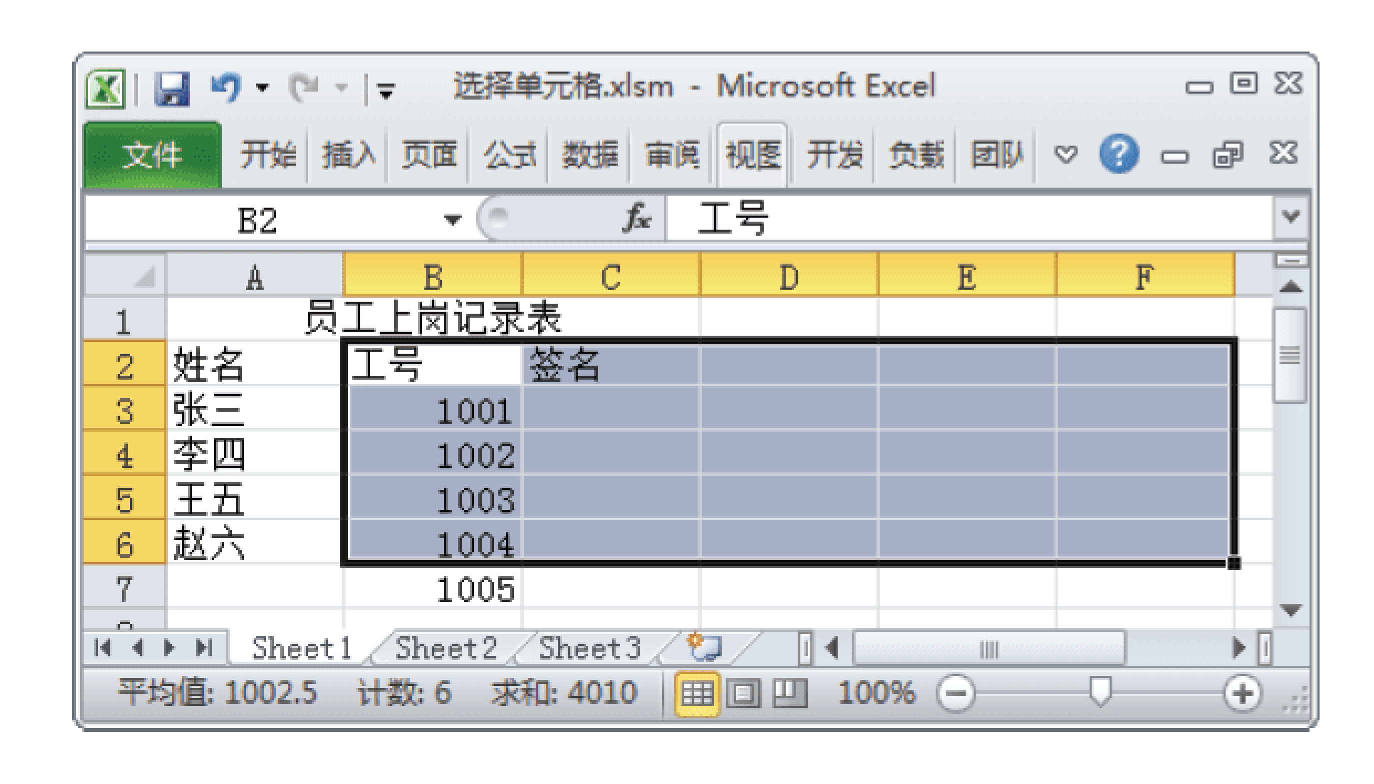


图 13.4　扩展单元格区域

【范例 13-2】 编写程序，搜索指定分数统计表中“总分”列单元格中的重复数据，相同数据显示其地址，代码如下所示。

```
01  Sub 搜索相同数据()
02      Dim r As Range                    '定义单元格变量
03      Set r = Range("F2:F6")            '指定变量为名为“总分”的行
04      For n = 1 To r.Rows.Count         '开始循环
05          If r.Cells(n, 1) = r.Cells(n + 1, 1) Then
                                          '判断相邻 2 个单元格数据是否相等
06              MsgBox "在单元格" & r.Cells(n, 1).Address& "和" & r.Cells(n
                + 1, 1). _
07              Address &"出现重复数据！" '相等则显示地址
08          End If
09      Next
10  End Sub
```

【运行结果】在 Excel 中选择工作表“总分”所在的列中的非空单元格，命名为“总分”。创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序对表中名为“总分”的列的数据进行比较，发现相同数据即给出提示，如图 13.5 所示。

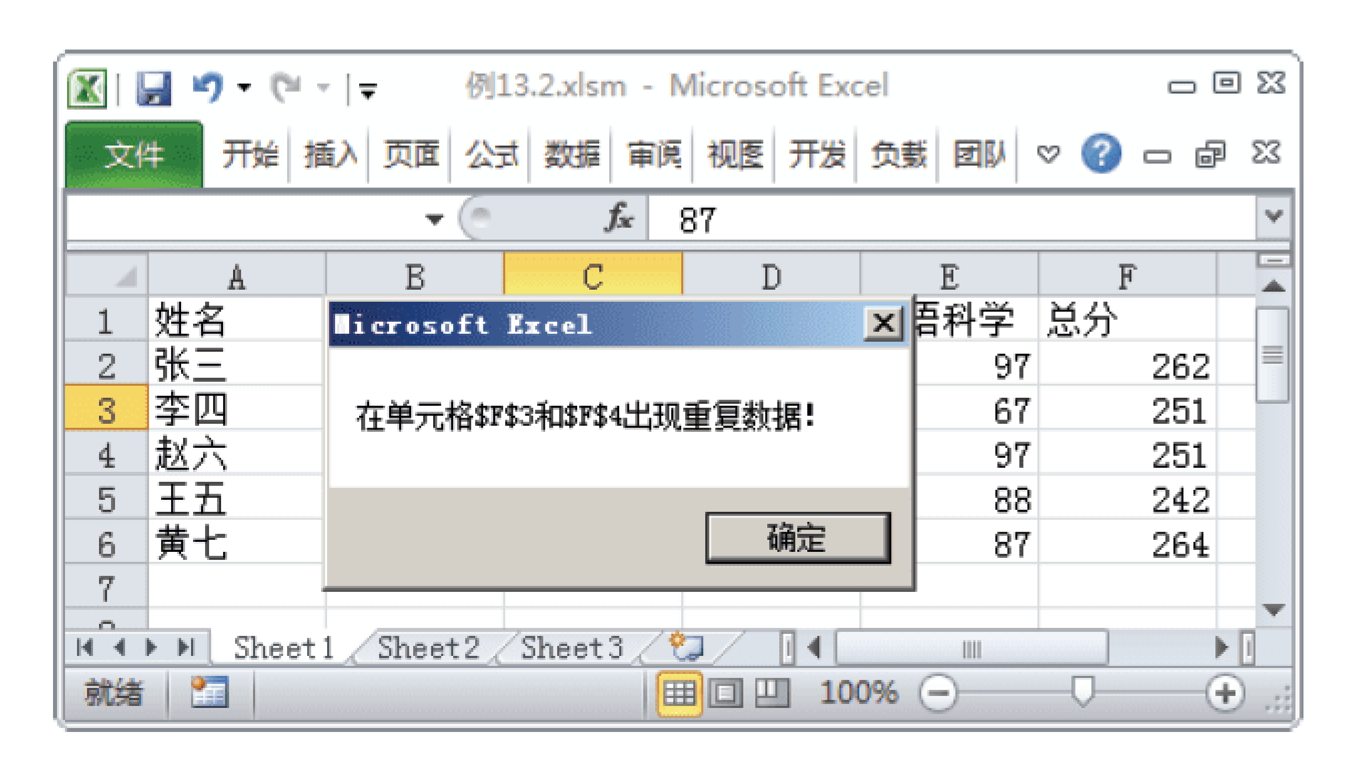


图 13.5　程序运行效果

【代码解析】本范例实现对某一列数据的重复数据进行搜索。程序中使用 For…Next 循环来遍历“总分”行所在的单元格，将相邻 2 个单元格数据进行比较，相同时则显示它们的地址。在程序的第 03 行将名为“总分”的单元格赋予单元格变量，第 04 行使用 Rows 属性获得“总分”单元格所在的行，使用 Count 方法来计算行中单元格的个数。在第 05 行，由于 r 对象是一个只有一列的单元格对象，因此使用 Cells 属性来获得单元格对象时，其列号为 1。

注意：程序第 03 行的“总分”不是列标题文字，而是图 13.2 中框选的单元格的名称，初学者要注意不要搞混。这个名称需要在程序运行前指定，否则程序会出错。

13.1.3　使用偏移方式引用 Excel 单元格

使用偏移方式来选择单元格，指的是以某个单元格或单元格区域为基准，通过给出朝某个方向的相对变化值来确定单元格的位置，这个变化值称为偏移量。在程序中，使用 VBA 的 Range 对象的 Offset 属性能够获得相对于指定单元格区域一定的偏移量位置上的区域。

Offset 属性的语法个数如下所示：

```
表达式.Offset(RowOffset, ColumnOffset)
```

参数说明：

- ❑ RowOffset 参数。区域偏移的行数，其值可为正数、负数或 0。其中，正数表示向下偏移，负数表示向上偏移。参数的默认值是 0。
- ❑ ColumnOffset 参数。区域偏移的列数，其值可为正数、负数或 0。其中，正数表示向右偏移，负数表示向左偏移。参数的默认值是 0。

【范例 13-3】 完成数据输入后横向选择单元格，代码如下所示。

```
Private Sub Worksheet Change(ByVal Target As Range)
    Target.Offset(0, 1).Select              '选择同行右侧单元格
End Sub
```

【运行结果】在 Visual Basic 编辑器的工作表“代码”窗口中输入上面的事件代码。在 Excel 工作表中输入数据，按 Enter 键后，自动选择右侧单元格，如图 13.6 所示。

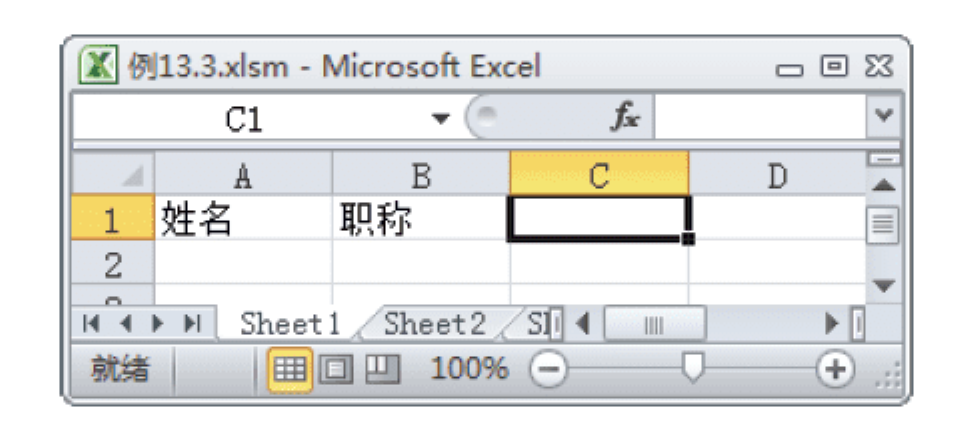


图 13.6　程序运行效果

【代码解析】在本范例使用 Worksheet 的 Change 事件来实现程序的功能。在程序中通过 Target 参数来获得数据发生改变的单元格对象，使用 Offset 属性引用该单元格右侧的单元格，使用 Select 方法来选择该单元格。

13.2　操作 Excel 单元格

单元格的操作包括单元格的复制和插入以及单元格的删除和隐藏等操作，这些操作都是使用程序进行数据处理时的基本操作。本节将介绍利用 VBA 程序来实现这些操作的方法。

13.2.1　删除 Excel 单元格

在 VBA 中，删除单元格可以使用 Delete 方法来实现，其语法格式如下：

```
表达式.Delete(Shift)
```

在这里，参数 Shift 仅用于 Range 对象，其指定如何调整单元格以替换删除的单元格。该参数可以取值为 xlShiftToLeft，表示右侧单元格向左移动；也可取值 xlShiftUp，表示下方的单元格向上移动。如果省略此参数，Microsoft Excel 将根据区域的形状确定调整方式。

【范例 13-4】 编写程序删除工作表中的空白行，代码如下所示。

```
01  Sub 删除空白行()
02      Dim n As Long                                       '声明变量
03      For n = Cells(1048576, 1).End(xlUp).Row To 1 Step -1  '遍历所有行
04          If Cells(n, 1) = "" Then                        '如果单元格为空
05              Cells(n, 1).EntireRow.Delete                '删除整行
06          End If
07      Next
08  End Sub
```

【运行结果】启动 Excel，创建一个学生成绩表，成绩表中存在着空白行，如图 13.7 所示。切换到 Visual Basic 编辑器，在工程资源管理器中创建一个模块，打开该模块的“代码”窗口，输入上述代码。按 F5 键运行程序，工作表中的空白区域被删除，如图 13.8 所示。

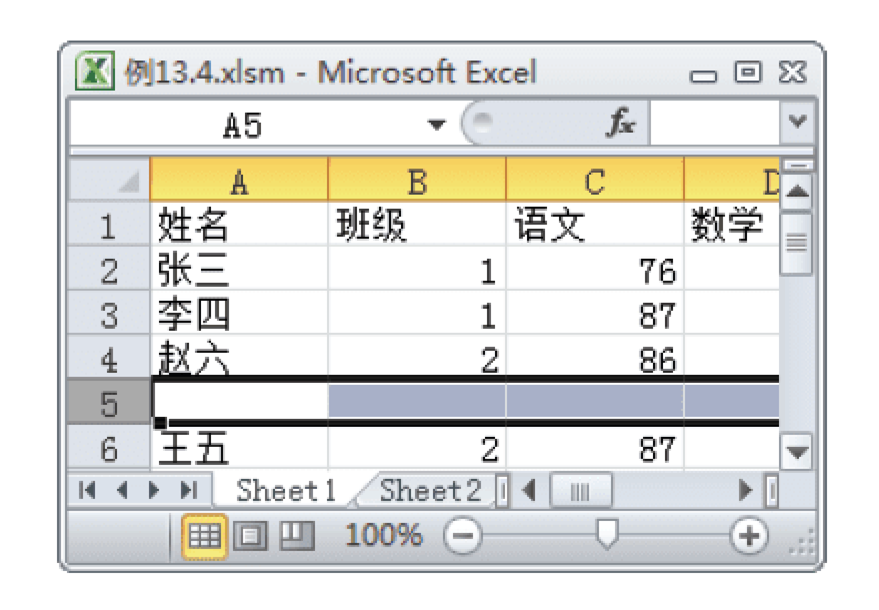


图 13.7　工作表中存在空白行

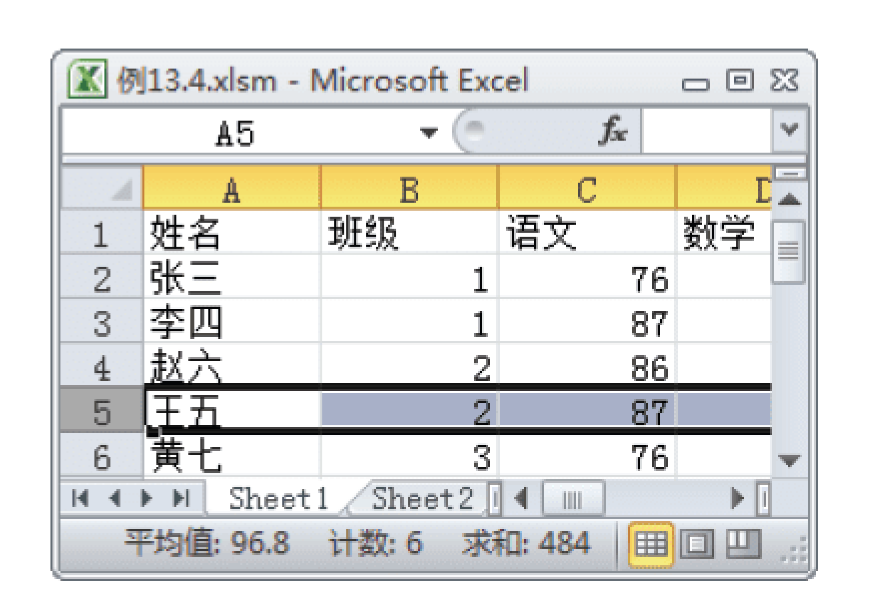


图 13.8　删除空白行后的效果

【代码解析】本范例使用 Delete 方法来删除单元格区域，程序的关键是确定查找整个单元格中的空白行。在第 03 行代码中 Cells(1048576, 1).End(xlUp).Row 语句获取第 1 列中非空行数，Cells(1048576, 1).End(xlUp)表示第 1 列最后一个单元格向上的第一个非空单元格。Row 属性将返回区域的第一行的行号，在本例中就是最后一个单元格的行号。第 04~06 行判断单元格是否为空，如果为空，则删除该单元格所在行。

提示： Excel 相对于以前的版本，单元格的数量大大增加，其最大行数为 1048576 行，Cells(1048576,1)可表示第一列最后一个单元格。

13.2.2　清除 Excel 单元格内容

清除单元格信息包括清除单元格内容和清除单元格格式。清除单元格的内容是将单元格变为空白，但单元格的格式将保留。清除单元格格式是在保留单元格内容的情况下将针对单元格所有格式设置清除掉。

在 VBA 中，使用 Clear 方法将能够清除单元格的内容以及格式设置，其语法格式如下所示：

```
对象.Clear
```

在 VBA 中，如果需要清除单元格的格式设置，可以使用 ClearFormat 方法来实现，而如果需要只清除单元格的内容，则可以使用 ClearContents 方法。如果需要删除单元格中注释，可以使用 ClearComments 方法。

【范例 13-5】 删除工作表数据区中的数据，保留格式和注释等信息行，代码如下所示。

```
01  Sub 删除工作表数据()
02      Range("C2: F60").SpecialCells(xlCellTypeConstants).Select
                                                        '选择使用过的单元格
03      Selection.ClearContents                         '清除内容
04  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，工作表中数据被清除，但批注、公式等信息仍然保留，如图 13.9 所示。

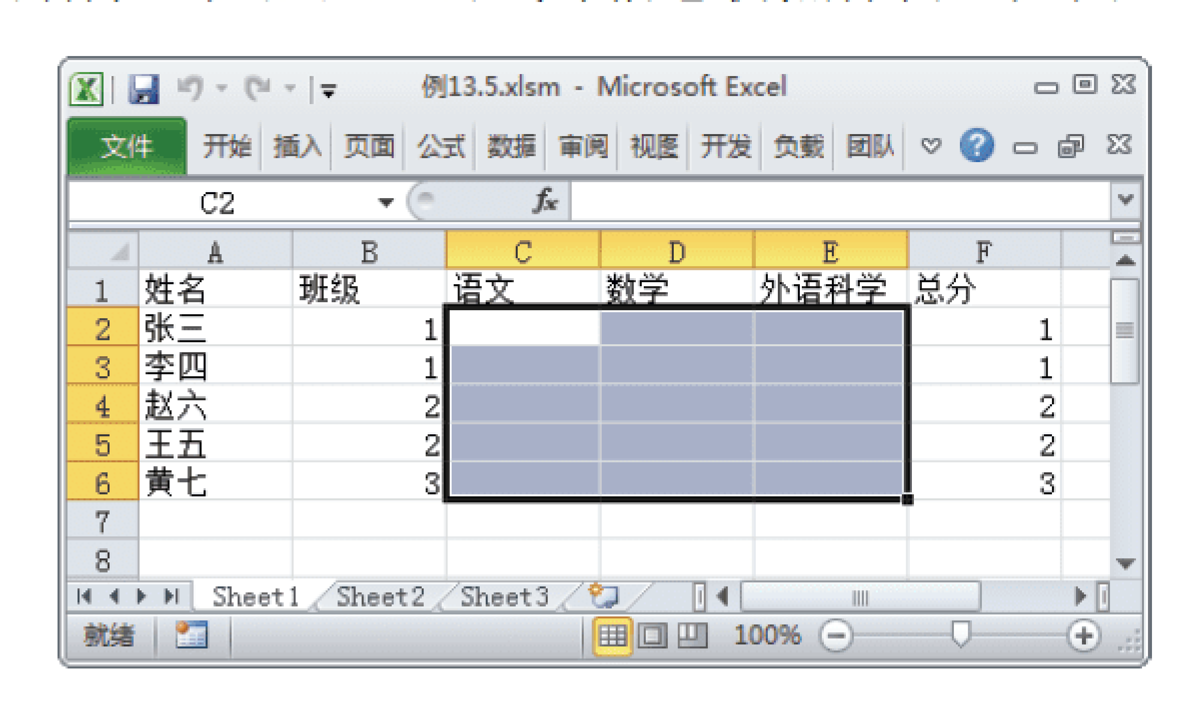


图 13.9　程序效果

【代码解析】本范例用于清除指定区域单元格的数据。在代码中，指定清除数据的区域为“C2”至“F60”区域，使用 ClearContents 方法清除了这个选择区域的内容。在程序的第 02 行中，.SpecialCells(xlCellTypeConstants)用来选择数据区作为清除内容的单元格区域。

13.2.3　插入和隐藏 Excel 单元格

在工作表中插入行或列可以使用 Insert 方法，其可以在工作表中插入单元格或者一个单元格区域。在使用 Insert 方法插入单元格时，将移动其他单元格以获得相应的空间。Insert 方法的语法格式如下所示：

```
表达式.Insert(Shift, CopyOrigin)
```

参数说明：

- Shift 参数用于指定单元格的调整方式。该参数是 xlInsertShiftDirection 常量之一，其中，xlShiftToRight 常量表示右移单元格，xlShiftDown 常量表示下移单元格。如果省略此参数，Microsoft Excel 将根据区域的形状确定调整方式。
- CopyOrigin 参数用于指定复制的起点。

如果需要隐藏工作表的某个单元格，可以将该单元格的 Hidden 属性设置为 True。如，隐藏工作表的 A 列，可以使用下面语句：

```
Worksheets("Sheet1").Columns("C").Hidden = True
```

注意：在使用 Hidden 属性时，指定的单元格区域必须占据整个行或整个列。

【范例 13-6】 在指定位置插入空行和空列，代码如下所示。

```
Sub 插入单元格()
    Dim myRange As Range                                '声明变量
```

```
03     Set myRange = Range("B2")                     '指定单元格对象
04     With myRange
05         MsgBox "在指定单元格上方插入一行"           '提示插入位置
06         .EntireRow.Insert Shift:=xlShiftDown      '在指定单元格上方插入一行
07         MsgBox "在基准单元格左边插入一列"           '提示插入位置
08         .EntireColumn.Insert Shift:=xlShiftToRight
                                                     '在指定单元格左边插入一列
09      End With
10  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序提示行和列插入的位置，如图 13.10 所示。完成后在工作表中插入一行空白行和一行空白列，如图 13.11 所示。

【代码解析】本程序演示使用 Insert 方法在工作表中插入空行和空列的方法。程序的第 03 行定义单元格对象。在第 06 行程序中，使用 EntireRow 属性获得指定单元格所在的行，使用 Insert 方法来实现插入一行的操作。插入时使用 Shift 参数指明行插入的位置。第 08 行使用 EntireColumn 获得单元格所在的列，在其左侧插入一个空白列。

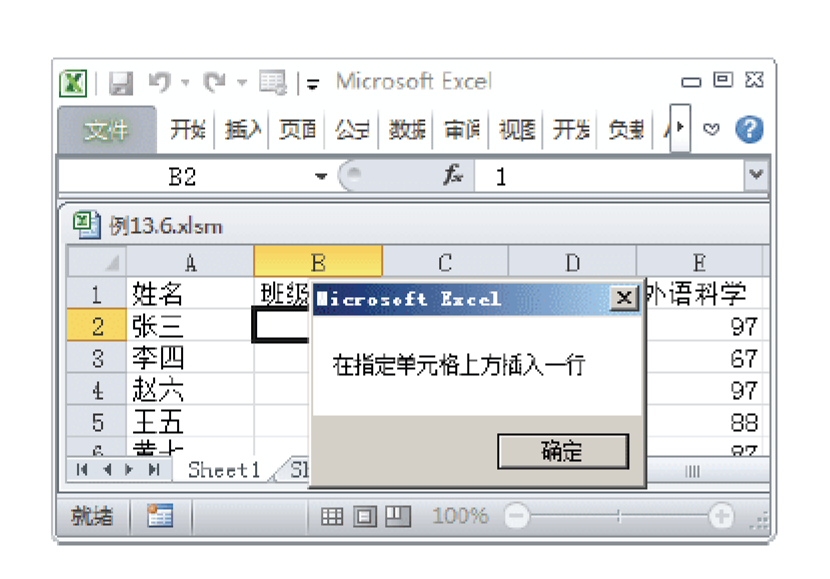


图 13.10　程序给出插入行提示

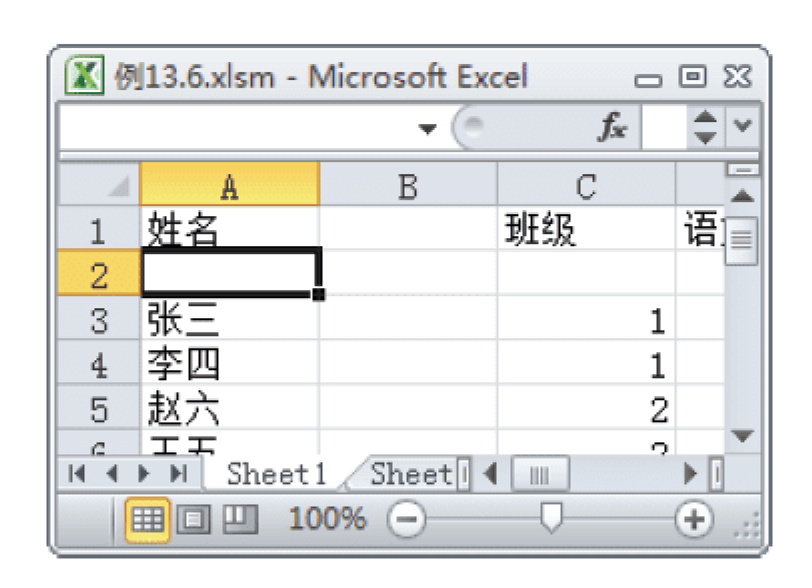


图 13.11　程序运行后插入空行、空列

13.2.4　复制 Excel 单元格数据

单元格间数据传递的一种方式是数据的复制和粘贴。在 VBA 中使用 Copy 方法能够实现单元格数据的复制，其语法格式如下：

```
表达式.Copy(Destination)
```

这里，Destination 参数用于指明复制的目标区域，如果此参数省略，表示将内容复制到剪贴板中。

如果需要将剪贴板上的内容粘贴到工作表中，可以使用 Paste 方法来实现，其语法格式如下：

```
表达式.Paste(Destination, Link)
```

参数说明：

- Destination 参数用于指明一个 Range 对象，指定用于粘贴剪贴板中内容的目标区域。如果省略此参数，就使用当前的选定区域。仅当剪贴板中的内容能被粘贴到某区域时，才能指定此参数。如果指定了此参数，则不能使用 Link 参数。
- Link 参数如果值为 True，则链接到被粘贴数据的源。指定此参数后，Destination

参数将不可用。此参数默认值是 False。

注意：如果不指定 Destination 参数，则在使用该方法之前必须选择目标区域。该方法可能会修改工作表的选定区域，这取决于剪贴板中的内容。

【范例 13-7】 将工作簿中 Sheet2 工作表中指定单元格的内容复制到 Sheet1 工作表中，复制完成后清除 Sheet2 工作表的内容，代码如下所示。

```
01   Sub 复制单元格()
02       Worksheets("Sheet2").Range("A2:B8").Copy            '复制单元格内容
03       Worksheets("Sheet1").Paste Destination:= _
04       Worksheets("Sheet1").Range("A9:D30")                '复制单元格到指定位置
05       Worksheets("Sheet2").Cells.Clear                    '清除复制后单元格内容
06   End Sub
```

【运行结果】在工作簿的 Sheet1 工作表中创建表格，如图 13.12 所示。切换到 Visual Basic 编辑器，在工程资源管理器中创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，“Sheet2”工作表中的数据被复制到“Sheet1”工作表已有数据的后面，同时“Sheet2”工作表中的数据被清除。单元格复制前后效果对比如图 13.13 所示。

【代码解析】本程序演示单元格复制和粘贴的方法，使用本范例的程序可以快速实现工作表数据的追加。在程序中，第 02 行复制指定工作表中的单元格区域，在第 03 行将复制到剪贴板中的数据粘贴到“Sheet1”工作表中，使用 Destination 参数指定数据粘贴的位置。第 05 行将粘贴后原工作表中的数据清除。

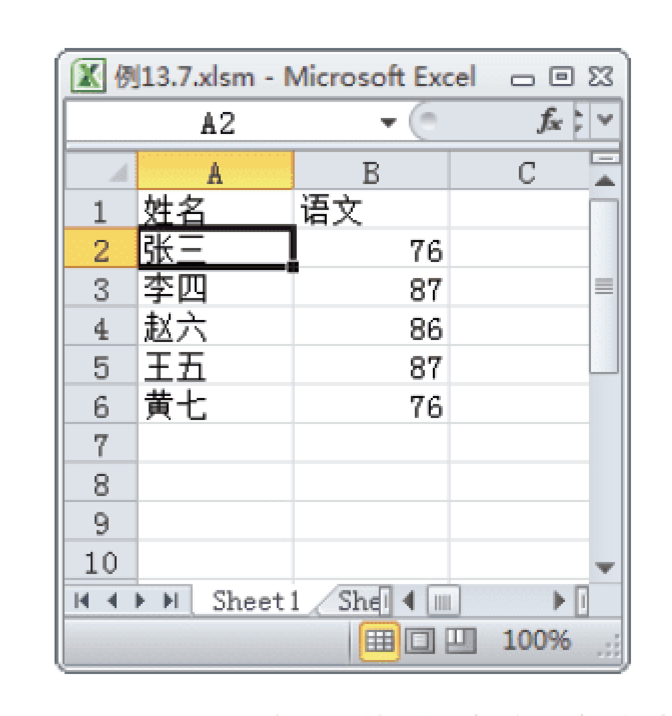




图 13.12　在工作表中创建表格

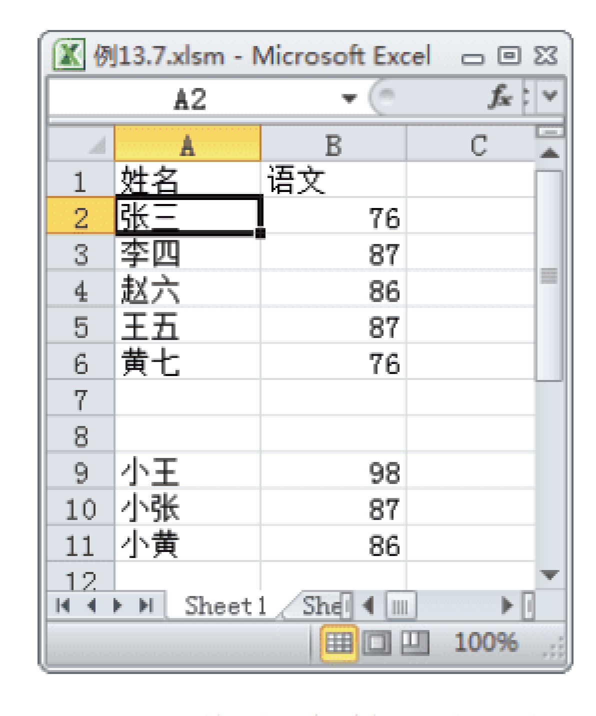


图 13.13　将 Sheet2 工作表中数据复制到 Sheet1 工作表中

13.2.5　保护 Excel 单元格

为了保护工作表中某些重点数据，可以将数据所在的单元格锁定。锁定后的单元格将无法进行任何更改，例如，无法在锁定的单元格中插入、修改和删除数据，也无法对单元格的格式进行修改。单元格的锁定，可以使用 Range 对象的 Locked 属性来实现，例如，锁定 A1 至 C5 单元格，可以使用下面的语句来完成：

```
Range("A1:C5").Locked=True
```

提示：如果需要解除单元格的锁定，只需要将 Locked 的属性设置为 False 即可。

在工作表中，有时需要对单元格中的公式进行保护，此时可以使用单元格的

FormulaHidden 属性将公式隐藏。如，隐藏 E1～E10 单元格中的公式，可以使用下面的语句：

```
Range("E1:E10").FormulaHidden=True
```

【范例 13-8】 编写程序，锁定工作表中所有带有公式的单元格，代码如下所示。

```
Sub 锁定单元格()
    Dim rng As Range                          '声明变量
    ActiveSheet.Unprotect                     '解除工作表保护
    For Each rng In ActiveSheet.UsedRange     '遍历当前工作表中所有单元格
       If Not rng.HasFormula Then             '如果不含有公式
          rng.Locked = False                  '单元格解除锁定
       Else
           rng.Locked = True                  '单元格锁定
       End If
    Next
    ActiveSheet.Protect                       '启用工作表保护
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行后修改带有公式的单元格，Excel 会禁止操作并给出提示，如图 13.14 所示。而工作表中其他数据区域可以任意修改。

【代码解析】本程序可以实现对工作表中带有公式的单元格区域的保护。程序首先使用 Unprotect 方法取消对工作表的保护以便重新对单元格进行锁定。程序使用 For Each…In 结构遍历工作表中所有使用的单元格，以实现对这些单元格的操作。在循环体中，首先判断单元格是否是带有公式的单元格，使用 HasFormula 属性值来进行判断，当其属性值为 True 时，表示单元格带有公式。对于带有公式的单元格将 Locked 属性设置为 True 锁定它，否则将 Locked 属性设置为 False 解除其锁定。最后，使用 Protect 方法重新启动对工作表的保护。

注意：只有在工作表被保护时，锁定单元格才会有效。工作表被保护时，默认的状态下，单元格是被锁定的。因此，在程序中需要对没有公式的单元格解除锁定，在程序开始要解除工作表保护，同时在完成设置后需要重新启动工作表保护。

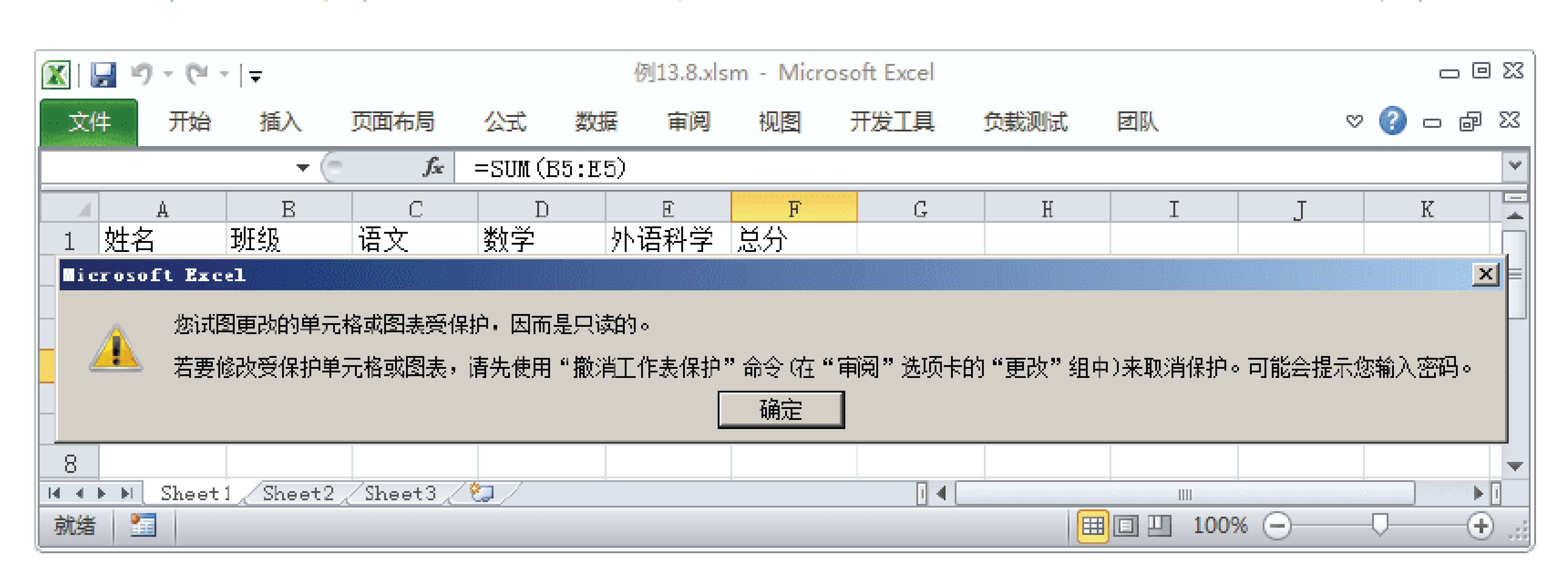


图 13.14　修改单元格后的提示

13.3　查找单元格数据

在进行数据的处理时，常常需要快速定位某个数据，这就是数据查找的问题。有时也需要对特定数据进行筛选，即从工作表中获得只满足条件的数据。Excel VBA 提供了用于

数据查找和筛选的方法，能够轻松实现工作表中数据的查找和筛选。

13.3.1　查找单个条件的数据

在 Excel VBA 程序中，可以使用 Find 方法来实现在工作表中对特定数据的查询，该方法的语法格式如下所示：

```
表达式.Find(What, After, LookIn, LookAt, SearchOrder, SearchDirection,
MatchCase, MatchByte, SearchFormat)
```

参数说明：

- What 为要搜索的数据，该参数可为字符串或任意 Microsoft Excel 数据类型。
- After 表示搜索过程将从其之后开始进行的单元格，此单元格对应于从用户界面搜索时的活动单元格的位置。
- LookIn 为信息类型。
- LookAt 可为以下 XlLookAt 常量之一，xlWhole 或 xlPart。
- SearchOrder 可为以下 XlSearchOrder 常量之一，xlByRows 或 xlByColumns。
- SearchDirection 为搜索的方向。
- MatchCase。如果为 True，则搜索区分大小写。默认值为 False。
- MatchByte 只在已经选择或安装了双字节语言支持时适用。如果为 True，则双字节字符只与双字节字符匹配。如果为 False，则双字节字符可与其对等的单字节字符匹配。
- SearchFormat 用于指定搜索的格式，Find 方法将能够找到符合该格式的数据。

注意：这里，After 参数必须是区域中的单个单元格，搜索是从该单元格之后开始的。如果不指定该参数，搜索将从区域的左上角的单元格之后开始。

【范例 13-9】 编写程序，查询工作表中“灯泡”的数量，代码如下所示。

```
Sub 查找文本()
    Cells.Find(what:="灯泡").Activate                    '查找灯泡
    MsgBox "灯泡本月销售个数为" & ActiveCell.Offset(0, 1).Value & "个"
                                                        '显示灯泡数量
    End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口，输入上述代码。按 F5 键运行程序，程序运行后，将在提示对话框中显示查找到的结果，如图 13.15 所示。

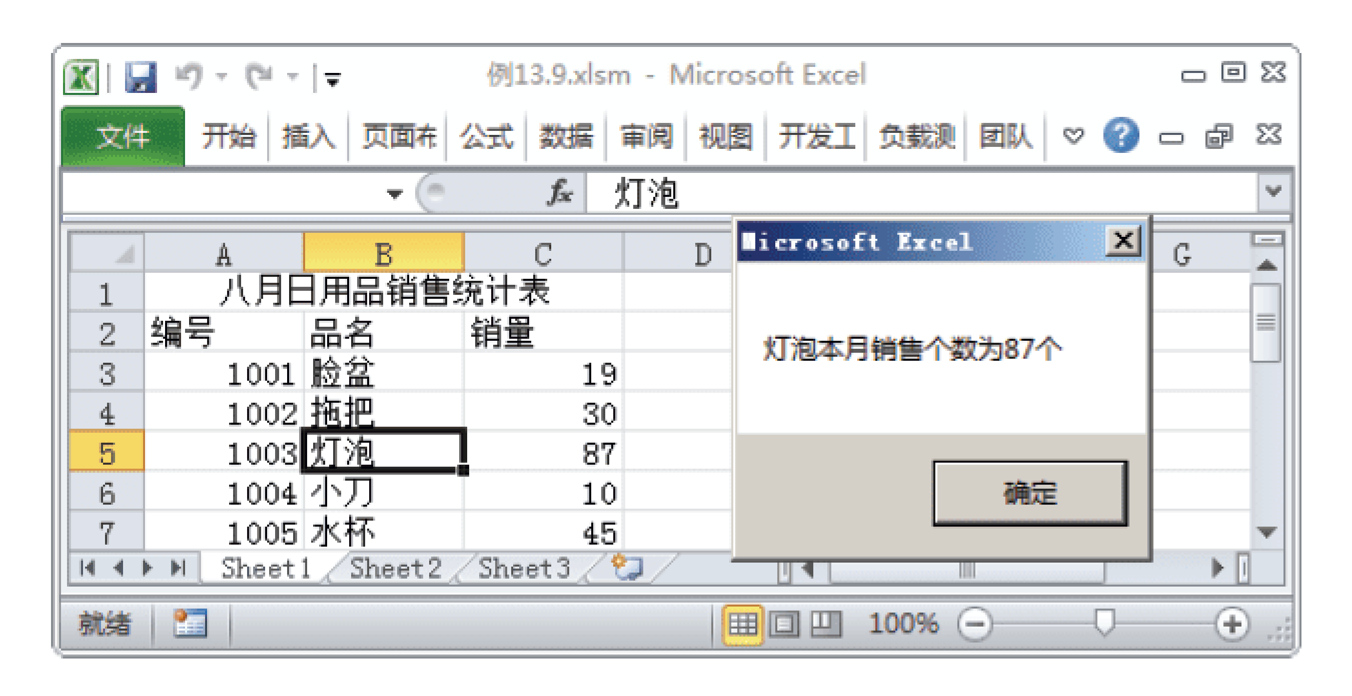


图 13.15　对话框中显示查找结果

【代码解析】本程序是使用 Find 方法查找数据的范例。在程序中以 What 参数指明 Find 方法查找的内容，使用 Activate 方法激活查找到的单元格。在显示查找结果时，由于需要显示的是灯泡的数量，因此使用 Offset(0,1)获得找到单元格右侧的单元格，这个单元格中的值就是灯泡的数量。

警告： 在使用 Find 方法时，如 What 参数指定的查找对象是不存在的文字，在程序运行时，Visual Basic 编辑器提示程序“运行时错误”。

13.3.2 查找多个条件的数据

使用 Find 方法找到第一个满足条件的数据后，不管该区域是否还有其他符合条件的数据，查找都将停止。VBA 提供了 FindNext 和 FindPrevious 方法来实现对区域中所有符合条件数据的查找。

FindNext 方法能够继续由 Find 方法开始的搜索，查找匹配相同条件的下一个单元格，并返回表示该单元格的 Range 对象，该操作不影响选定内容和活动单元格。FindNext 方法的语法格式如下所示：

```
表达式.FindNext(After)
```

这里，After 参数用于指定一个单元格，查找将从该单元格之后开始。此单元格对应于从用户界面搜索时的活动单元格位置。注意，After 必须是查找区域中的单个单元格，使用 FindNext 方法进行的搜索将从该单元格之后开始。如果未指定本参数，查找将从区域的左上角单元格之后开始。

FindPrevious 方法和 FindNext 方法语法格式相同，只是其为从指定单元格向前进行查找而已。

【范例 13-10】 编写程序，查找工作表中所有“灯泡”项目，找到的项目高亮显示，代码如下所示。

```
Sub 查找多个文本()
    Dim c As Range, m As Range, n As Integer              '声明变量
    Set c = Cells.Find(what:="灯泡")                      '查找“灯泡”
    c.Activate                                            '激活找到单元格
    c.EntireRow.Interior.Color = RGB(124, 100, 135)'填充单元格所在行颜色
    c.EntireRow.Font.ColorIndex = 45                      '改变文字颜色
        For n = 1 To Range("b500").End(xlUp).Row + 1'查找整个列的单元格
            Set c = Cells.FindNext(after:=ActiveCell)     '查找下一个目标
            c.Activate                                  '激活找到的单元格
            c.EntireRow.Interior.Color = RGB(124, 100, 135) '填充行颜色
            c.EntireRow.Font.ColorIndex = 45              '更改文字颜色
        Next
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口，输入上述代码。按 F5 键运行程序，程序中所有符合条件的行高亮显示，如图 13.16 所示。

【代码解析】本程序可用于查询数据表中某个数据的位置。在程序中，首先使用 Find

方法查询第一个符合条件的数据，并将其高亮显示。然后使用 FindNext 方法查询表中其他符合条件的数据。为了能够查询到所有的数据，使用 For…Next 语句来实现对工作表中所有非空单元格的查询。

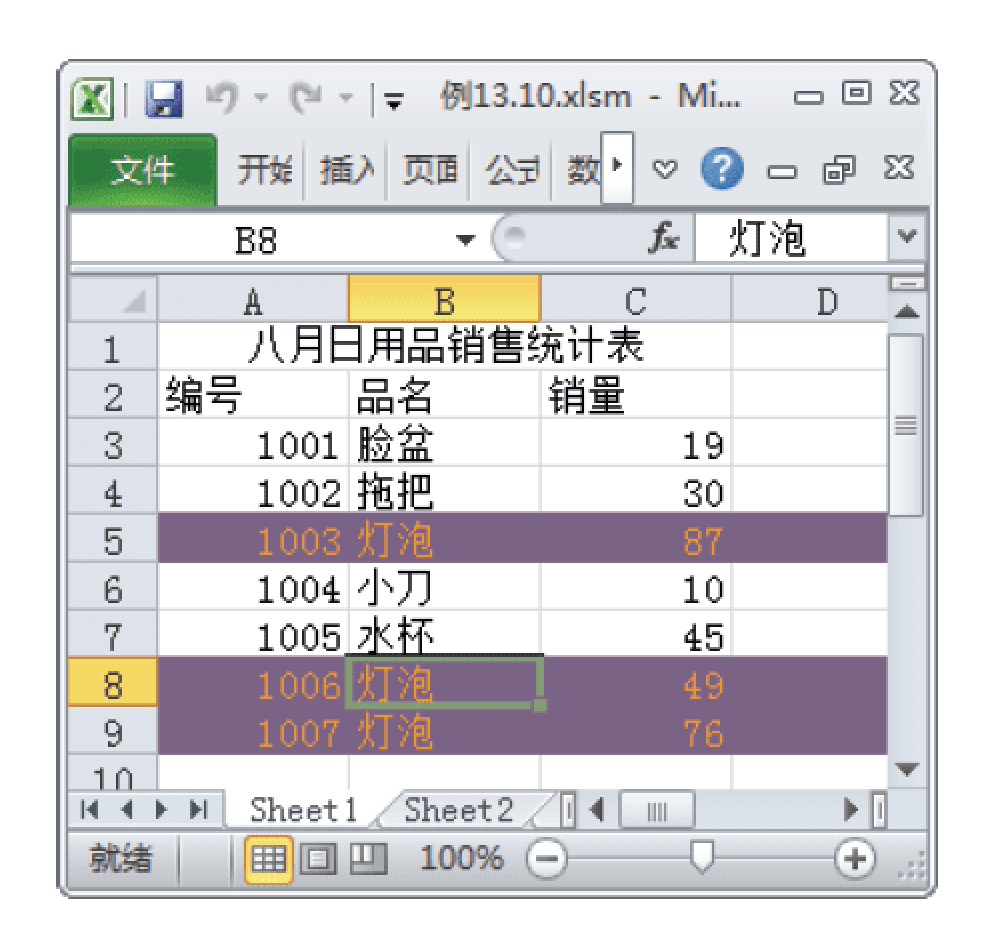


图 13.16　程序运行结果

在第 07 行代码中，使用 Range("b500").End(xlUp).Row 获取 B 列非空单元格的数量，之所以要将其加 1，是为了避免最后一行的数据没有查询的情况发生。在第 09 行代码中使用 Activate 方法激活符合条件的单元格。由于 FindNext 方法设置的是从激活单元格向后查找，因此找到的符合条件的单元格后必须要激活它。程序的第 05、06 行和第 10、11 行都是使用 EntireRow 属性获得单元格所在的行，对整行单元格进行设置。

提示：由于符合条件的数据可能不止一个，要想对整个区域进行查找，一般都需要使用循环语句来进行反复查找。

13.3.3　使用 Match 方法查找数据

Match 是 WorksheetFunction 对象方法，其能够返回指定方式下与指定数值相匹配数值在数组中的位置。在编写 VBA 程序时，使用该方法能够更方便地在数据表中查找需要数据的位置，同时还能引用查找到的数据。Match 函数的语法格式如下：

```
表达式.Match(Arg1, Arg2, Arg3)
```

参数说明：

- Arg1 指明需要在表中查找的值。
- Arg2 指明可能包含所要查找的值的连续单元格区域，其值必须为数组或数组引用。
- Arg3 的值可为数字-1、0 或 1，用于指明 Microsoft Excel 如何将上面 2 个参数指定的值进行匹配。如果值为 1，查找小于或等于 Arg1 的最大数值；如果值为 0，则查找等于 Arg1 的值；如果值为-1，则查找大于或等于 Arg1 的最小值。

注意：在 Arg3 确定比较方式时，如果值为 1，Arg2 的数据必须按照升序排列。如果值为–1，Arg2 的数据必须按照降序排列。

【范例 13-11】 在成绩表中，按姓名查询某个学生的成绩，代码如下所示。

```
Sub 按人名查找()
    Dim name As String, n As Integer, sh As Worksheet       '声明变量
    On Error Resume Next                                     '错误处理
    name = InputBox("请输入需要查询的学生姓名", "学生成绩查询") '输入学生姓名
        If Not (name = Empty) Then                           '如果输入学生姓名
            For Each sh In Sheets                            '遍历所有单元格
                n = WorksheetFunction.Match(name, sh.Columns(1), 0)
                                                             '查找相同姓名
                    If Not (n = Empty) Then                  '找到相同姓名
                      MsgBox "姓名: " & name & Chr(13) _
                      & "班级: " & sh.Cells(n, 2) & Chr(13) _
                      & "语文: " & sh.Cells(n, 3) & Chr(13) _
                      & "数学: " & sh.Cells(n, 4) & Chr(13) _
                      & "外语: " & sh.Cells(n, 5) & Chr(13) _
                      & "科学: " & sh.Cells(n, 6) & Chr(13) _
                      & "总分: " & sh.Cells(n, 7)    '显示相关信息
                    Else
                      MsgBox "学生" & name &"不在表中! "
                                                    '没有找到学生给出提示
                    End If
                Next
            End If
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序给出输入对话框，要求输入需查询的学生姓名，如图 13.17 所示。输入学生姓名后，程序给出查询结果，如图 13.18 所示。

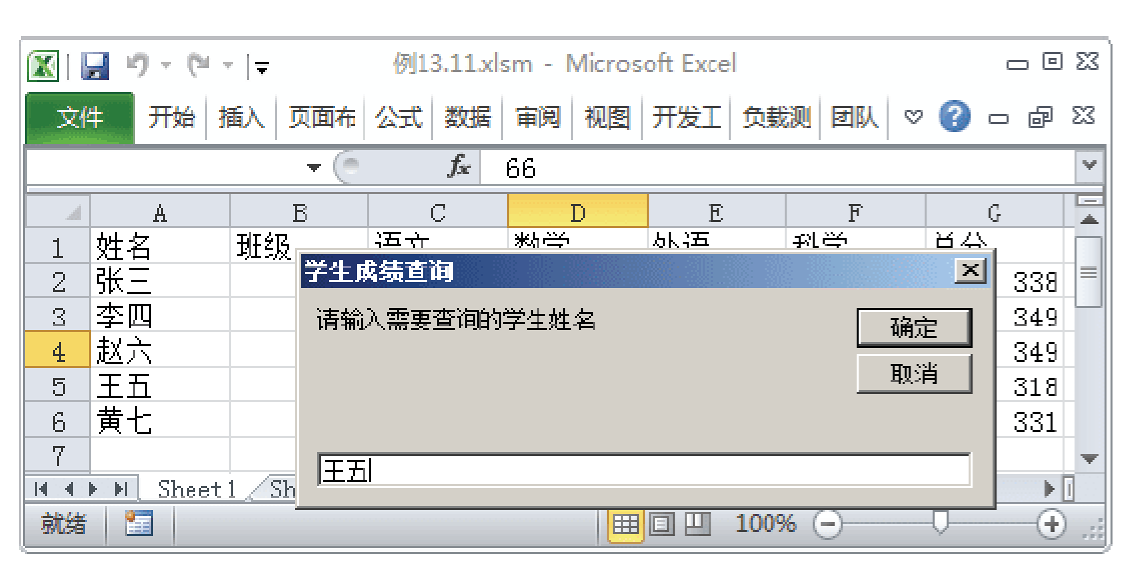




图 13.17　输入对话框输入需查询学生姓名

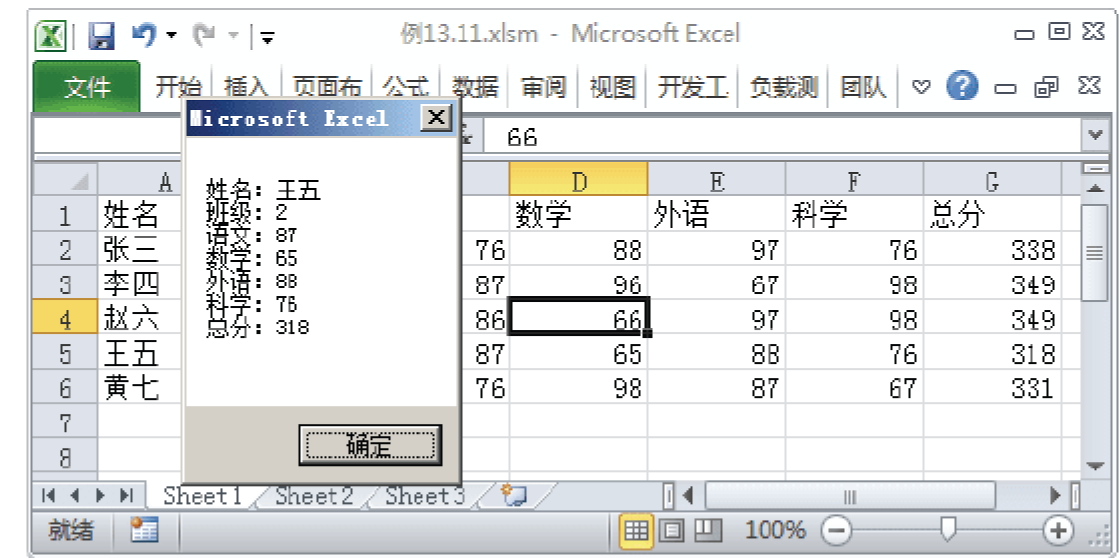


图 13.18　给出学生成绩

如果没有查到需要的学生，程序给出没有查询到的提示，如图 13.19 所示。

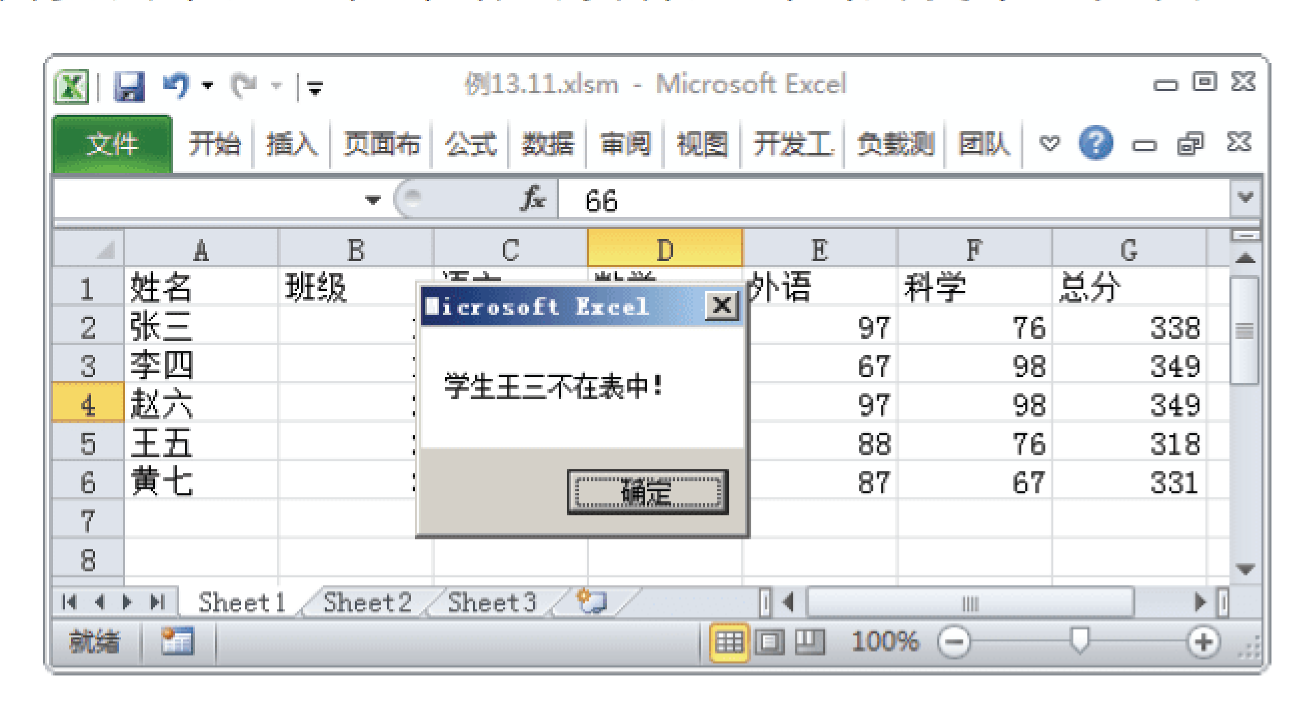


图 13.19　显示没有查询到提示

【代码解析】本程序实现在工作表中循环查找数据。程序运行时，首先获得用户输入姓名，在程序的第 05 行，以变量 name 是否为空来判断是否输入了姓名，为空则退出程序；不为空则在工作表中查询学生姓名。第 07 行使用 Match 方法在第 1 列中寻找与 name 变量值相同的内容，找到的位置值赋予变量 n。第 08 行判断是否找到了相匹配的内容，如果没有找到，则给出学生信息。由于 Match 方法能够给出找到的姓名在 sh.Columns(1)中的位置值，所以也可以使用 Cells(n,1)来引用单元格。

13.3.4 筛选符合条件的数据

数据的筛选是将满足条件的数据显示出来而将不符合条件的数据隐藏，可以说筛选实际上就是比查找多了一步数据显示的过程。在 VBA 中对数据的筛可以选择自动筛选和高级筛选这 2 种方式。

要实现对数据的自动查找，可以使用 Range 对象的 AutoFilter 方法，其语法格式如下：

```
表达式.AutoFilter(Field, Criteria1, Operator, Criteria2, VisibleDropDown)
```

参数说明：

- ❑ Field 为相对于作为筛选基准字段（从列表左侧开始，最左侧的字段为第一个字段）的字段的整型偏移量。
- ❑ Criteria1 为第一个筛选条件。如果省略该参数，则搜索条件为 All。如果将 Operator 设置为 xlTop10Items，则 Criteria1 指定数据项个数（例如“10”）。
- ❑ Operator 用于指定筛选类型，其值为一个 xlAutoFilterOperator 常量。
- ❑ Criteria2 为第二个筛选条件（一个字符串）。与 Criteria1 和 Operator 参数一起组合成复合筛选条件。
- ❑ VisibleDropDown 如果为 True，则显示筛选字段的自动筛选下拉箭头。如果为 False，则隐藏筛选字段的自动筛选下拉箭头。默认值为 True。

如：在一个学生成绩表中，学生姓名放置于成绩表的第一列。现在需要从该表的 A1 单元格开始筛选出一个清单，该清单中显示姓名为“李历”的数据项，同时在工作表第一行字段名处显示下拉箭头，如图 13.20 所示。

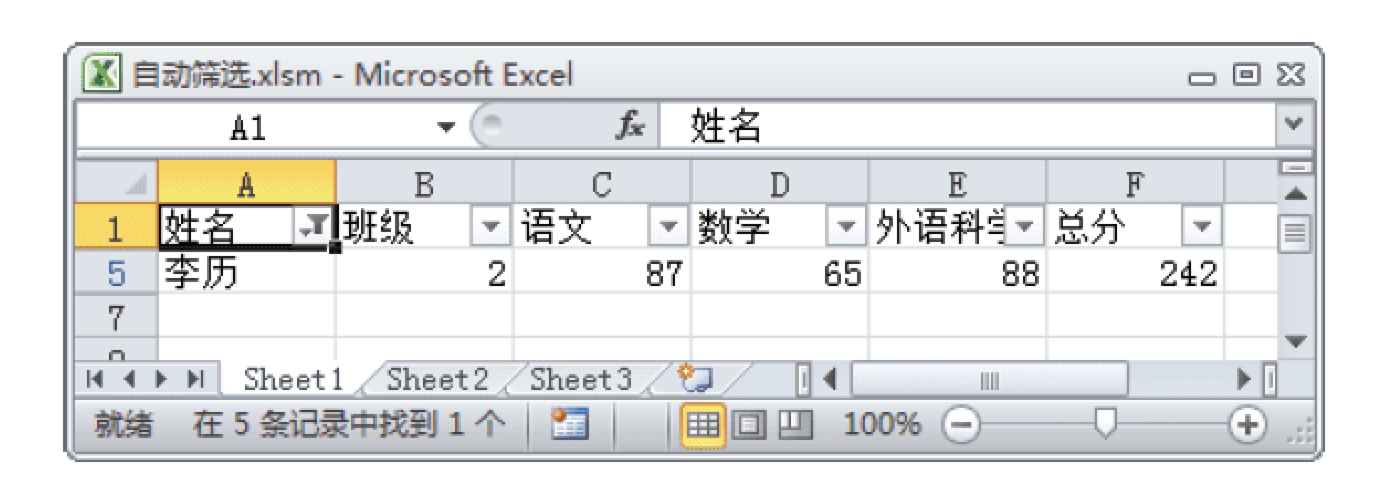


图 13.20 自动筛选的结果

此时可使用下面的语句：

```
ActiveSheet.Range("A1").AutoFilter field:=1, Criteria1:="李历",_
VisibleDropDown:=True
```

如果需要用户自定义筛选的条件，可以使用高级筛选的方法。在程序中实现高级筛选，可以使用 Range 对象的 AdvancedFilter 方法，其语法格式如下所示：

```
表达式.AdvancedFilter(Action, CriteriaRange, CopyToRange, Unique)
```

参数说明：

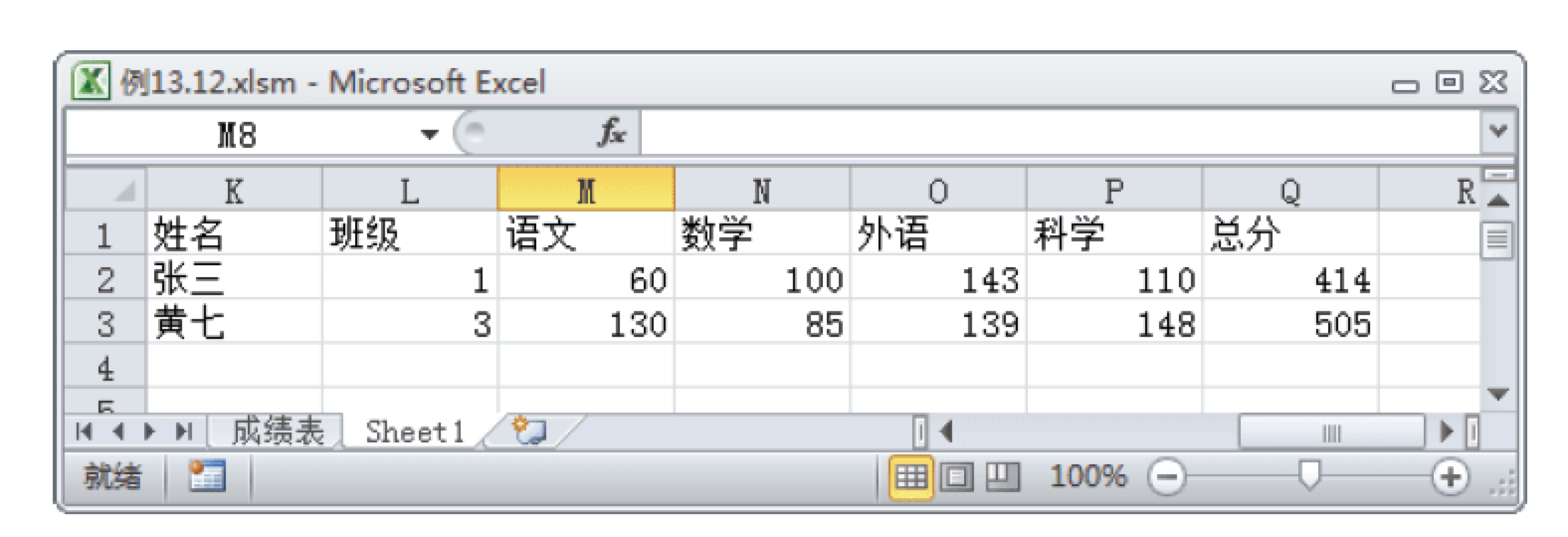
- Action 为 XlFilterAction 的常量之一，用于指定是否就地复制或筛选列表。
- CritenriaRange 为条件区域。如果省略该参数，则没有条件限制。
- CopyToRange。如果 Action 为 xlFilterCopy，则复制行的目标区域；否则，忽略该参数。
- Unique。如果为 True，则只筛选唯一记录；如果为 False，则筛选符合条件的所有记录。参数的默认值为 False。

【范例 13-12】 在成绩表中，查询总分高于 400 分的学生的成绩，将这些学生的成绩放入新的单元格中，代码如下所示。

```
Sub 查找特定分数的学生()
    ActiveSheet.Range("a1:g50").AdvancedFilter Action:=xlFilterCopy, _
    CriteriaRange:=Range("i1:i2"), CopyToRange:=Worksheets("sheet1")
    .Range("k1:q50"), _
    Unique:=False                    '查找总分高于 400 分的学生
End Sub
```

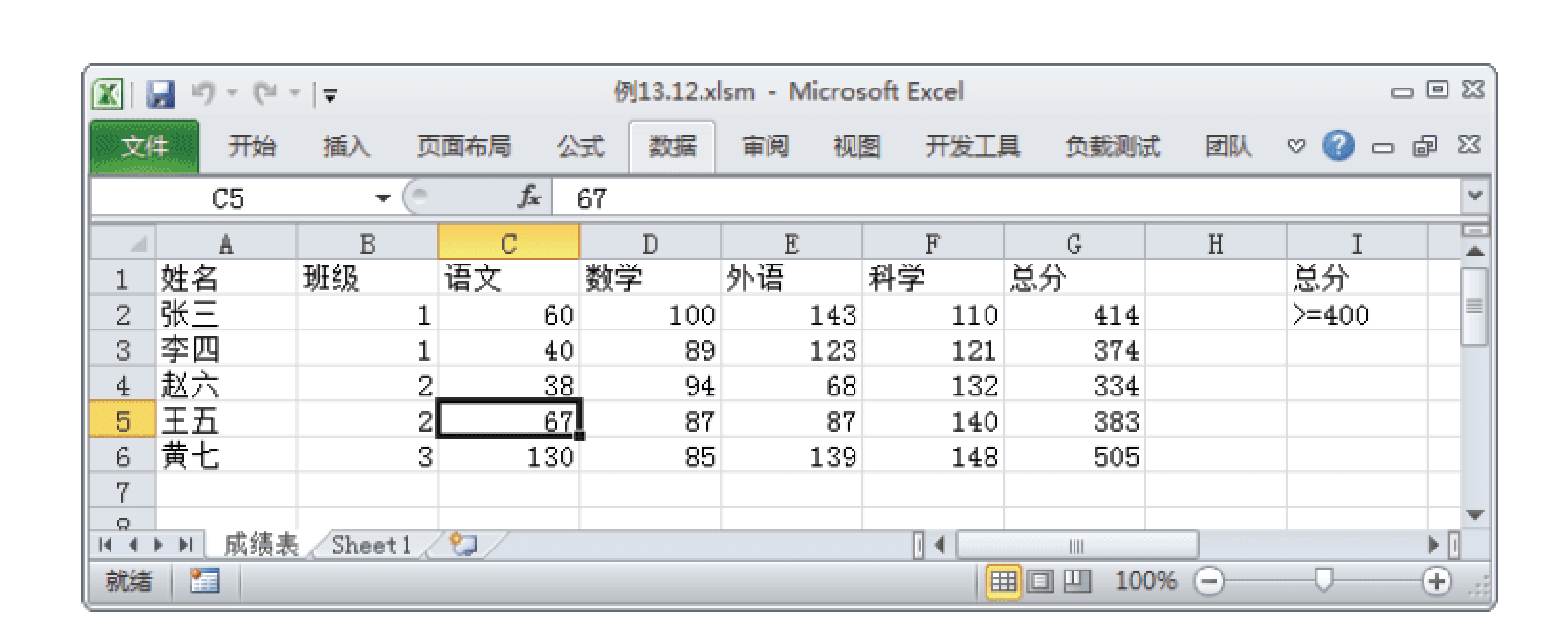
【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行效果如图 13.21 所示。

【代码解析】本程序将筛选特定条件的数据并将获得的数据复制到一个空白工作表中。程序运行前，需要在工作表中创建筛选条件，如图 13.22 所示。



	K	L	M	N	O	P	Q
1	姓名	班级	语文	数学	外语	科学	总分
2	张三	1	60	100	143	110	414
3	黄七	3	130	85	139	148	505

图 13.21　将筛选的成绩复制到新的工作表中



	A	B	C	D	E	F	G	H	I
1	姓名	班级	语文	数学	外语	科学	总分		总分
2	张三	1	60	100	143	110	414		>=400
3	李四	1	40	89	123	121	374		
4	赵六	2	38	94	68	132	334		
5	王五	2	67	87	87	140	383		
6	黄七	3	130	85	139	148	505		

图 13.22　创建筛选条件

程序使用 AdvancedFilter 方法进行数据筛选，其中 Range("a1:g50")指明数据查询范围，Action:=xlFilterCopy 表示复制查询结果，CriteriaRange:=Range("i1:i2")指定查询条件在数据表中的位置，CopyToRange:=Worksheets("sheet1").Range("k1:q50")将查询到的结果复制到

K1 至 Q50 单元格区域中，Unique:=False 表示筛选出符合条件的所有的记录。

提示：参数 Action 可用的常量有 2 个，当其设置为 xlFilterCopy 时，筛选结果复制到新位置。当其设置为 xlFilterInPlace 时，保留数据不动。

13.3.5　按颜色筛选数据

在 Excel 中，为了清楚地表示不同的数据，往往在创建表格时会使用不同的颜色来标记不同类型和用途的数据。在 Excel 中，通过 VBA 编程，能够筛选出这些特殊的数据。

在 Excel VBA 中，颜色有 2 种表现方式，它们是 RGB 颜色和调色板颜色。其中，RGB 颜色由红、绿和蓝 3 种基本颜色构成，屏幕上每个像素的最终显示的颜色是由这 3 种颜色混合而成。在 VBA 中，可以使用 RGB 函数来设置对象的颜色值，其语法如下所示：

```
对象.颜色属性=RGB（参数 1,参数 2,参数 3）
```

这里，使用 RGB 函数为一个对象的颜色参数赋值，函数的参数 1、参数 2 和参数 3 分别表示颜色中的红色成分、绿色成分和蓝色成分，它们的值均在 0～255 之间。

在 VBA 中，调色板是一个有 56 个颜色项的 RGB 颜色列表，颜色表中的颜色通常采用索引号来表示，因此调色板能够提供的颜色又常常称为索引颜色。使用这种方法来进行颜色的设置能够大大减小程序对内存的需求。VBA 系统默认的调色板及其对应的索引颜色值如图 13.23 所示。

在 VBA 中，可以使用索引号为对象的 ColorIndex 属性赋值，实现对象颜色的设置。同时，读取该属性的值，也能够获得颜色的索引号。在制作工作表时，如果使用颜色对这些项目进行了标示，则在对数据进行筛选时，可以通过查询索引颜色值来获取这些特殊的数据。下面通过一个具体的范例来介绍根据颜色筛选数据的方法。

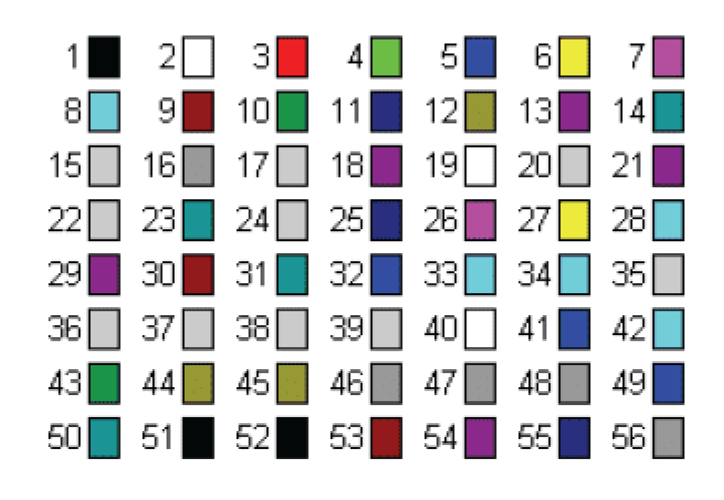


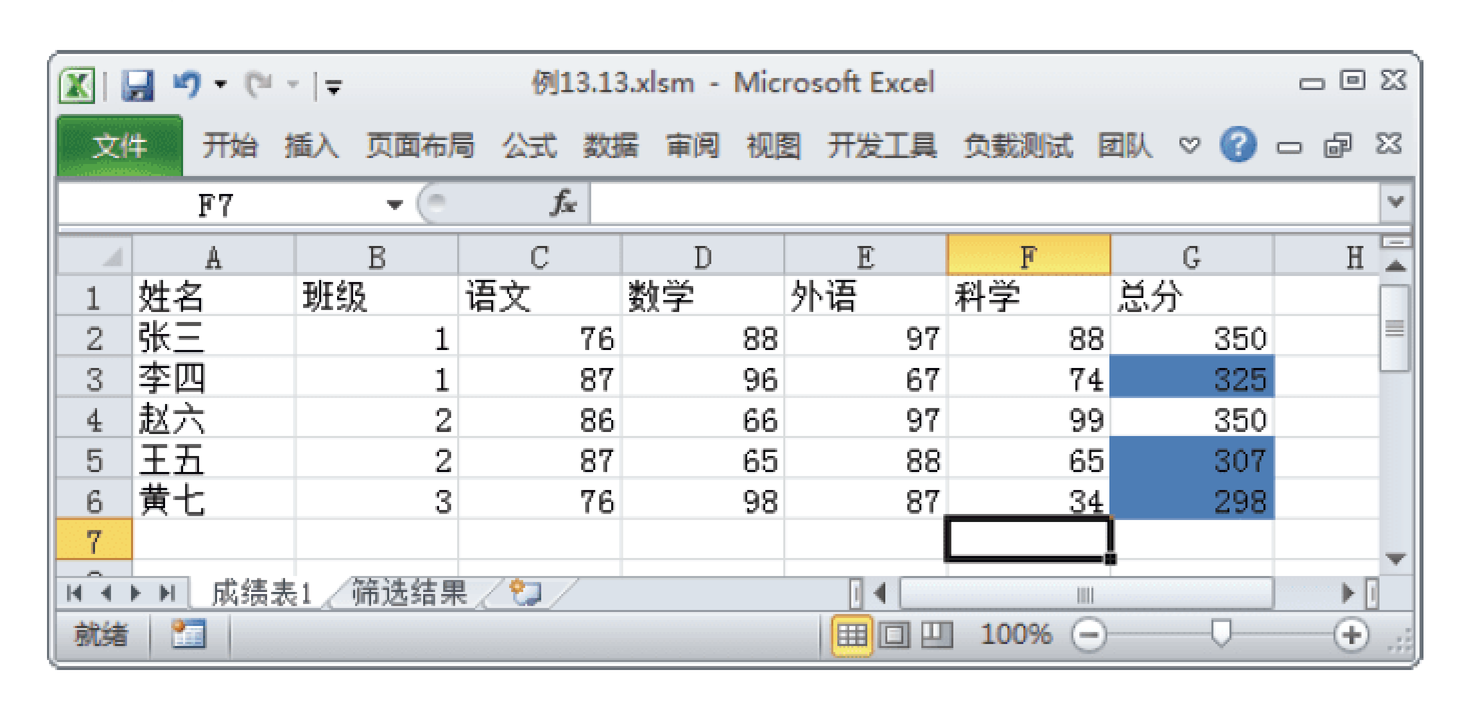
图 13.23　系统默认调色板及其对应的颜色索引号

【范例 13-13】 在名为“成绩表 1”的工作表中，总分列中某些单元格被填充了颜色，将这些单元格对应的学生成绩记录复制到名为“筛选结果”工作表中，代码如下所示。

```
Sub 通过颜色筛选数据()
        Dim rng As Range, i As Long, n As Integer        '声明变量
        Dim p As Long, q As Integer
        n = 1                                            '变量初始化
        q = Sheets("成绩表 1").Range("G2").Interior.ColorIndex'获取颜色值
        Application.ScreenUpdating = False               '关闭屏幕更新
        p = Sheets("成绩表 1").Range("a800").End(xlUp).Row
                                              '获取 A 列最后一个非空单元格行号
```

```
08          For Each rng In Sheets("成绩表1").Range("G2:G" & p)
                                          '遍历工作表 G 列所有单元格
09              If rng.Interior.ColorIndex = q Then'如果单元格填充颜色相同
10                  i = rng.EntireRow.Row             '获取单元格所在行的行号
11                  n = n + 1                 '变量加 1
12                  Rows(i).Copy Sheets("筛选结果").Rows(n)
                                          '将整行复制到“筛选结果”工作表中
13              End If
14          Next
15          Application.ScreenUpdating = True         '恢复屏幕更新
16          Sheets("筛选结果").Activate               '激活“筛选结果”工作表
17  End Sub
```

【运行结果】打开学生成绩表，成绩表中某些学生的总分被填充了颜色进行标示，如图 13.24 所示。创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行效果如图 13.25 所示。



	A	B	C	D	E	F	G
1	姓名	班级	语文	数学	外语	科学	总分
2	张三	1	76	88	97	88	350
3	李四	1	87	96	67	74	325
4	赵六	2	86	66	97	99	350
5	王五	2	87	65	88	65	307
6	黄七	3	76	98	87	34	298

图 13.24　某些学生总分被填充了颜色

	A	B	C	D	E	F	G
1							
2	张三	1	76	88	97	88	350
3	李四	1	87	96	67	74	325
4	黄七	3	76	98	87	34	298

图 13.25　按颜色筛选的结果

【代码解析】本示例演示筛选被颜色标注的单元格数据的方法。由于工作表中，总分列的 G2 单元格即是一个被填充了颜色的单元格，程序的第 05 行获取该单元格填充颜色的颜色索引号。在第 08～14 行的 For Each…In Next 循环中，遍历总分列中的每一个单元格，将单元格的 ColorIndex 属性值与 G2 单元格的颜色索引号进行比较。如果相同，则使用 i=rng.EntireRow.Row 语句获得该单元格的行号，在第 12 行中使用该行号引用整列，并使用 Copy 方法将该行的内容复制到“筛选结果”工作表中由变量 n 指定的行中。

提示： 通过本范例可以看到，按照颜色筛选数据，实际上与普通数据的筛选方式是一样的，无非是将查询条件变成了单元格填充颜色的索引颜色值，即 ColorIndex 属性值。

13.4　设置 Excel 单元格格式

设置单元格指的是进行与单元格外观有关的设置，其中包括设置单元格边框、定义单元格文字字符的格式以及设置条件格式等。

13.4.1　设置 Excel 单元格边框

Excel 的单元格区域的边框包括左右边框、顶部边框及底部边框这 4 个分立的边框，VBA 可以通过 Borders 对象来实现对边框的控制。单元格的各个边框构成 Borders 集合，各个边框可以单独返回，也可以作为一个组返回。

Range 对象的 Borders 属性能够返回一个 Borders 对象集合，如果要分别控制边框线，可以使用下面语句来实现：

```
Borders(Index)
```

这里，Index 用于控制边框线的类型，当其值设置为 xlDiagonalDown 时，表示从区域中每个单元格的左上角至右下角的边框。当其值设置为 xlDiagonalUp 时，表示从区域中每个单元格的左下角至右上角的边框。当其值设置为 xlEdgeLeft、xlEdgeRight、xlEdgeTop 或 xlEdgeBottom 时，分别表示区域左侧、右侧、顶部和底部的边框。当其值为 xlInside Horizontal 或 xlInsideVertical 时，表示区域中所有单元格的水平边框或垂直边框（区域以外的边框除外）。

设置边框线型是通过设置 Borders 对象的 LineStyle（即线型）属性来实现。LineStyle 属性可用的常数一共有 8 个，包括 xlContinuous、xlDash、xlDashDot 和 xlDot 等，它们分别对应实线、虚线、点划线和点式线等。

设置边框的线宽可以通过设置 Borders 对象的 Weight 属性来实现。该属性可以设置为 4 个常量中的一个，它们分别是 xlHairline、xlThin、xlMedium、xlThick。这 4 个常量分别表示特细、细、中等宽度和粗。

【范例 13-14】 将工作表的 A1～E1 单元格合并为一个单元格，编写程序为单元格添加左右边框和下边框并且将边框设置为红色的粗实线，代码如下所示。

```
Sub 设置边框()
     With Worksheets("Sheet1").Range("A1:E1").Borders(xlEdgeBottom)
                                                        '设置底部边框
         .LineStyle = xlContinuous                      '边框为实线
         .Weight = xlThick                              '设置为粗线
         .ColorIndex = 3                                '设置边框线颜色
    End With
    With Worksheets("Sheet1").Range("A1:E1").Borders(xlEdgeLeft)
                                                        '设置左侧边框线
         .LineStyle = xlContinuous
         .Weight = xlThick
         .ColorIndex = 3
    End With
    With Worksheets("Sheet1").Range("A1:E1").Borders(xlEdgeRight)
```

```
                                                    '设置右侧边框线
13              .LineStyle = xlContinuous
14              .Weight = xlThick
15              .ColorIndex = 3
16          End With
17      End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行效果如图 13.26 所示。

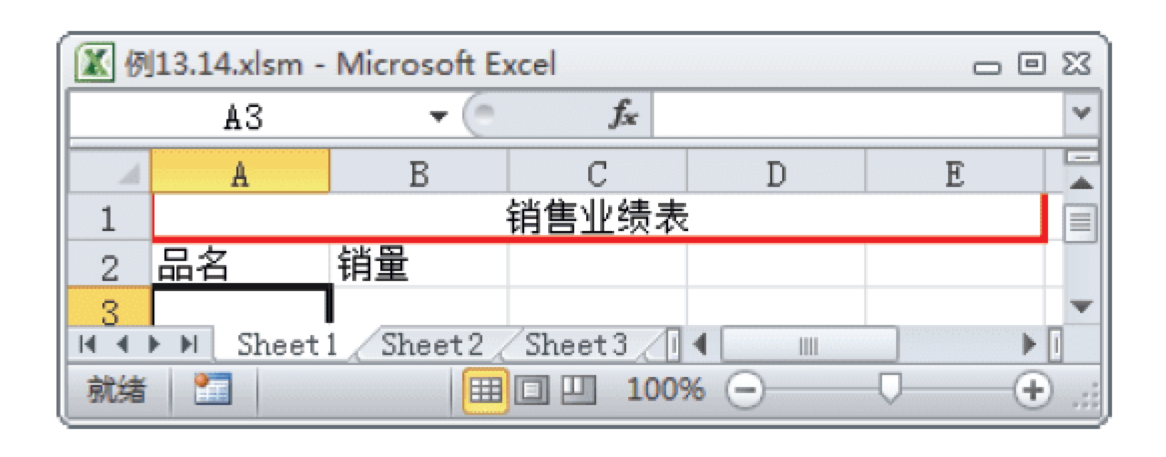


图 13.26　程序运行效果

【代码解析】本程序为指定的单元格设置左侧、右侧和底部边框。在第 02～06 行中，使用 Worksheets("Sheet1").Range("A1:E1").Borders(xlEdgeBottom)语句指明设置 A1～E1 单元格的边框底部边框。使用 With 结构对指定的单元格底部边框的线型、宽度和颜色进行设置。

注意：单元格的每一个边框都是一个对象，在对其进行设置时，只能分别进行设置，这是初学者在编写程序时应该注意的。

13.4.2　使用 Excel 条件格式

所谓的条件格式，指的是在单元格中的数据满足某个条件后，单元格将具有某种格式。Excel VBA 中使用 FormatCondition 对象来定义条件格式，多个条件格式对象构成条件格式集合 FormatConditions。使用 Range 对象的 FormatConditions 属性能够实现对 Format Conditions 对象集的引用。

要想向单元格中添加条件格式，可以使用 FormatConditions 对象集的 Add 方法，其语法格式如下所示：

```
表达式.Add(Type, Operator, Formula1, Formula2)
```

参数说明：

- Type 用于指定条件格式是基于单元格值还是基于表达式。
- Operator 为条件格式运算符，其可为下面的 xlFormatConditionOperator 常量之一，xlBetween、xlEqual、xlGreater、xlGreaterEqual、xlLess、xlLessEqual、xlNotBetween 或 xlNotEqual。如果 Type 为 xlExpression，则忽略 Operator 参数。
- Formula1 为与条件格式相关联的值或表达式，可为常量值、字符串值、单元格引用或公式。
- Formula2。当 Operator 为 xlBetween 或 xlNotBetween 时，它是与条件格式第二部分相关联的值或表达式（否则忽略该参数）。可为常量值、字符串值、单元格引用或公式。

🔔**提示**：对于 xlFormatCondition 的 Operator 常量，可以从其字面理解其含义，如 xlBetween 表示介于，xlEqual 表示等于，xlGreater 表示大于。

【范例 13-15】 在学生成绩表中，为总分高于工作表中 G10 单元格分数的单元格添加条件格式。单元格格式为红色边框和红色加粗数字，代码如下所示。

```
01  Sub 移动工作表示例()
02     With Worksheets(1).Range("G2:G50").FormatConditions _
03         .Add(xlCellValue, xlGreater, "=$g$10")    '设置条件格式
04         With .Borders
05            .LineStyle = xlContinuous              '设置边框线型
06            .Weight = xlThin                       '设置边框线为细线
07            .ColorIndex = 3                        '设置边框颜色
08        End With
09        With .Font
10         .Bold = True                              '文字设置为粗体
11         .ColorIndex = 3                           '设置文字颜色
12        End With
13     End With
14  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行程序，程序运行效果如图 13.27 所示。

例13.15.xlsm - Microsoft Excel

	A	B	C	D	E	F	G
1				学生成绩表			
2	姓名	班级	语文	数学	外语	科学	总分
3	张三	1	76	88	97	88	350
4	李四	1	87	96	67	74	325
5	赵六	2	86	66	97	99	350
6	王五	2	87	65	88	65	307
7	黄七	3	76	98	87	34	298
8	小华	3	88	87	65	78	321
9	小明	3	76	89	87	68	323
10	汤姆	4	89	81	76	88	338
11	伯恩	4	89	99	96	87	375

图 13.27 程序运行效果

【代码解析】本程序使用 FormatConditions 对象为单元格添加强调格式。在代码中 Range("G2:G50")指定需要添加格式的单元格，使用 FormatConditions 的.Add 方法来添加条件格式。这里，Add 方法中的参数 xlCellValue 表示条件是基于单元格的值， xlGreater 表示比指定单元格的值大，"=g10"表示以 G10 单元格的值作为条件。在程序的第 04～12 行，使用 With…End With 结构分别设置单元格边框和文字的样式。

🔔**提示**：With 结构是可以嵌套的，嵌套的 With 结构可以方便地实现同一个对象的不同类属性和方法的使用，这在编程中是提高编程效率的好办法。

13.4.3　使用 Excel 数据条

Excel 提供了一种新的条件格式设置方式，那就是数据条。使用数据条能够将数据的大小用颜色表示出来，能够使数据的显示更加直观。

使用数据条，首先要创建数据条对象，对象的创建使用 AddDatabar 方法，该方法是 FormatConditions 对象集的方法。完成 DataBar 对象的创建后，可以使用该对象的方法对数据进行设置。下面通过一个范例来看看数据条在数据表中的应用。

【范例 13-16】 使用数据条表示商品销售情况，代码如下所示。

```
01  Sub 数据条()
02      Range("b3:b15").FormatConditions.AddDatabar          '创建数据条对象
03      Range("b3:b15").FormatConditions(1).BarColor.ColorIndex = 10
                                                              '设置对象颜色
04  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口，输入上述代码。按 F5 键运行程序，表中单元格的数据添加了数据条，根据表中数据值的大小，数据条显示不同的长度，如图 13.28 所示。

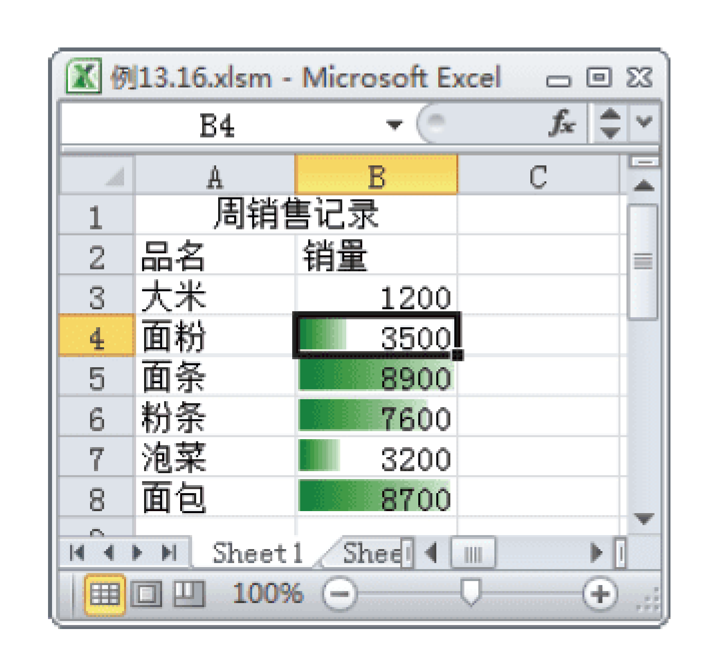


图 13.28　单元格中添加数据条

【代码解析】本范例程序介绍为单元格添加数据条和设置数据条样式的方法。程序的第 02 行为指定单元格区域中的对象创建数据条对象。第 03 行使用 FormatConditions(1)调用新创建的数据条对象，通过设置数据条对象的 BarColor 属性值来改变单元格中数据条显示的颜色。

13.4.4　自动排列前 10 名数据

在分析数据时，有时需要对数据按照大小进行排名，同时标示出排名前 10 位的数据。Excel VBA 的 FormatConditions 对象集中有一个 Top10 对象，该对象代表条件格式规则的前 10 项。在使用 Top10 对象时，可以先使用 AddTop10 方法创建对象获得符合条件的前 10 位单元格，然后使用对象的属性和方法来对获得的对象进行设置，为单元格添加格式。下面通过一个范例来介绍 Top10 对象的使用。

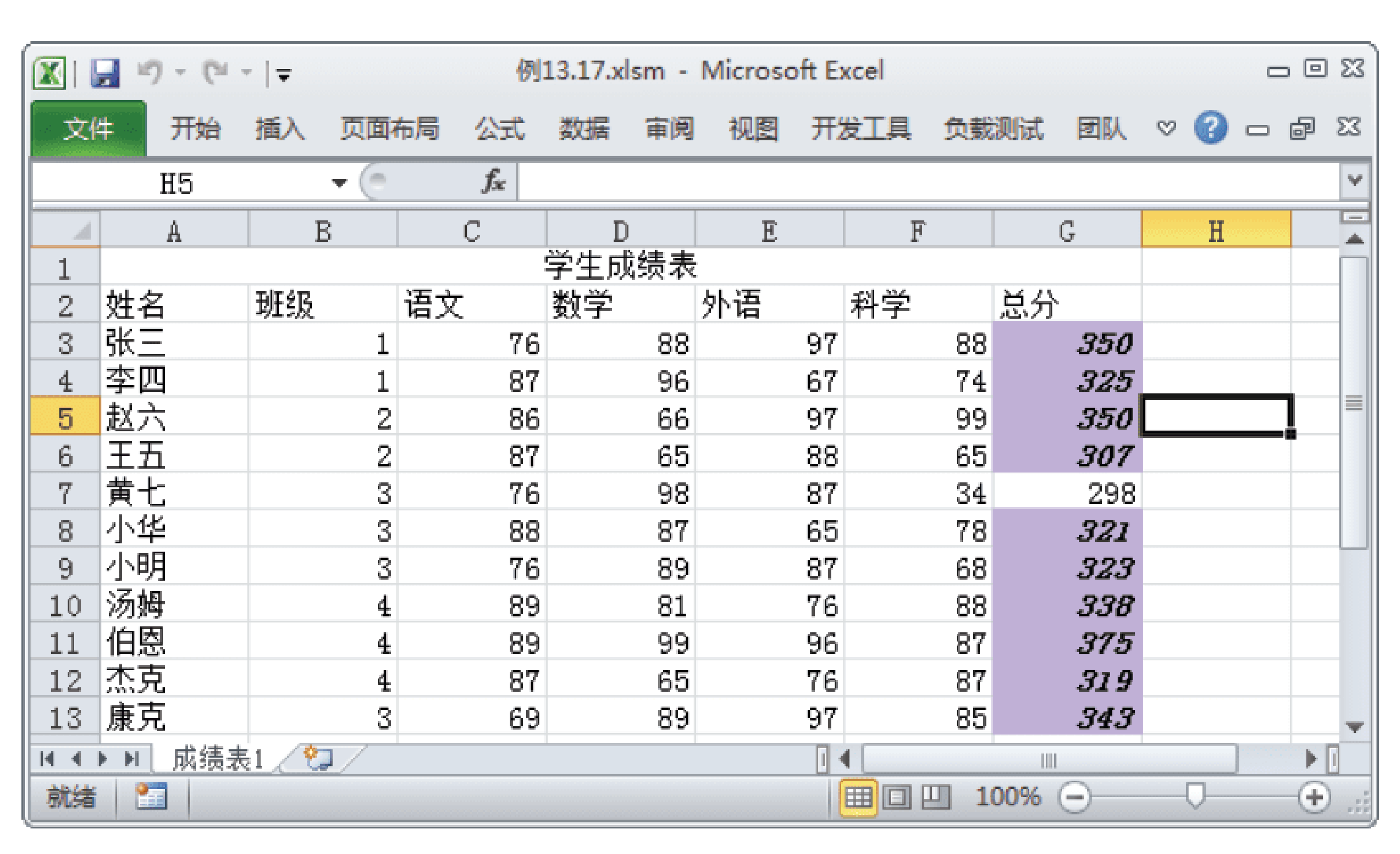
【范例 13-17】为工作表中前 10 名的总分添加特殊格式，代码如下所示。

```
Sub 自动标示前10名()
    Range("G2:G50").Select                             '指定单元格
    Selection.FormatConditions.AddTop10                '添加一个 Top10 对象
    Selection.FormatConditions(Selection.FormatConditions _
    .Count).SetFirstPriority                           '设置当前格式优先
    With Selection.FormatConditions(1)
        .TopBottom = xlTop10Top                        '从顶端排位
        .Rank = 10                                     '选取前 10 名
        .Percent = False                               '不使用百分比排列
    End With
    With Selection.FormatConditions(1).Font            '设置文字格式
```

```
12          .Italic = True                                    '使用斜体
13          .Bold = True                                      '文字加粗
14      End With
15      Selection.FormatConditions(1).Interior _
16       .Color = RGB(200, 180, 255)                          '设置填充颜色
17  End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口，输入上述代码。按 F5 键运行程序，程序指定的 G 列排名前 10 名的数据以特殊格式显示。由于篇幅限制，这里只显示部分特殊格式单元格的样式，如图 13.29 所示。

【代码解析】本范例程序介绍使用 Top10 对象为排名前 10 的数据添加特殊格式以突出显示的方法。在第 02 行代码中，Selection.FormatConditions.AddTop10 将返回代码符合指定区域条件格式规则的最高 10 个对象。在第 04 行中，Selection.FormatConditions.Count 获得条件格式对象集合的元素格式，然后使用 SetFirstPriority 方法将条件格式的优先级设置为第一优先级，这样能够保证在工作表上应用所有其他规则之前应用此规则。第 06～10 行代码使用 With 结构对条件规则进行设置。最后，使用 Font 属性和 Interior 属性来设置单元格的条件格式样式。



	A	B	C	D	E	F	G
1			学生成绩表				
2	姓名	班级	语文	数学	外语	科学	总分
3	张三	1	76	88	97	88	350
4	李四	1	87	96	67	74	325
5	赵六	2	86	66	97	99	350
6	王五	2	87	65	88	65	307
7	黄七	3	76	98	87	34	298
8	小华	3	88	87	65	78	321
9	小明	3	76	89	87	68	323
10	汤姆	4	89	81	76	88	338
11	伯恩	4	89	99	96	87	375
12	杰克	4	87	65	76	87	319
13	康克	3	69	89	97	85	343

图 13.29　程序运行效果

13.5　小　　结

单元格是 Excel 中最常用，也是最重要的对象，学习 Excel 程序开发必须掌握单元格对象的使用。掌握单元格、行、列和区域的引用是实现单元格操作必须掌握的基础技能。同时，单元格的基本操作包括单元格的添加、单元格的删除、单元格内容的清除、单元格的复制和粘贴以及单元格数据的保护等内容。同时，读者在学习本章后还应该掌握数据查找的常用方法，以及单元格格式的设置技巧。

单元格涉及的内容较多，单元格的操作与其他对象的联系较为紧密。除了掌握单元格对象的常见方法和属性外，读者还应该掌握其他相关对象的使用，如用于设置条件格式的

FormatConditions 对象、与文字相关的 Font 对象以及与边框有关的 Borders 对象等。熟练掌握常用对象的使用，是掌握 VBA 的必要条件。

13.6　本 章 习 题

1．下面引用单元格的方式哪一个是错误的？（　　）

A．Range("A1")　　B．Cells(1,"A")　　C．R[-2]C[3]　　D．"品名"

2．选择下面程序运行的结果。（　　）

```
Sub Test()
    Dim myRange As Range
    Sheet1.Activate
    Set myRange=Sheet1.Range(Cells(1,1),Cells(5,5))
    myRange=5
    Set myRange=Nothing
End Sub
```

A．工作表的 A1 单元格的值为 5　　B．工作表 A1 至 E5 单元格的值为 5

C．工作表所有单元格的值为 5　　D．工作表所有单元格被清空

3．选择下面程序的运行结果。（　　）

```
Sub Test ()
    Dim i As Long,n As Integer
    For n=1 To 3
        n=Selection.Row
        ActiveSheet.Rows(n).Insert
    Next
End Sub
```

A．在当前选择单元格下方插入一行　　B．在当前选择单元格下方插入 3 行

C．在当前选择单元格上方插入一行　　D．在当前选择单元格上方插入 3 行

4．在下面代码中使用 AdvancedFilter 方法来实现数据筛选，对该方法的描述正确的是哪项？（　　）

```
Sub test()
    Range("A1:G15").AdvancedFilter Action:=xlFilterCopy, _
    CriteriaRange:=Range("H1:H3"),CopyToRange:=Range("A17:G30"),
    Unique:=False
End Sub
```

A．程序错误无法执行　　B．筛选结果复制到 H1 至 H3 单元格中

C．筛选条件位于 H1 至 H3 单元格中　　D．筛选结果将不会被复制

5．编写程序，使数据表中不同分数段的成绩以不同颜色显示。

【提示】Excel 提供的设置条件格式的方法，只能允许同时使用 3 个条件，如果条件众多，则无法做到使用条件格式设置单元格样式。此时，可以通过条件判断语句来对条件进行判断，然后根据不同的结果来设置单元格的样式。

6．查找表格中的重复数据，并将重复数据使用虚线框标示，文字的颜色改为红色。

【提示】本练习的关键是如何获取相同的数据，这里不准备采取遍历单元格进行比较的方法。本练习使用 FormatConditions 对象的 AddUniqueValues 方法创建一个 UniqueValues 对象，该对象的 DupeUnique 属性能够确定条件格式的规则是查找区域中的重复值还是唯一值，将其设置为查找重复值，以此为条件即可向单元格添加条件格式，获得需要的结果。

第 14 章　工作表界面

界面是用户与应用程序交互的窗口，在交互界面上有各种各样的控件，例如输入数据的文本框、确认下一步的按钮等，Excel VBA 中的控件分为表单控件和 ActiveX 控件，常用的表单控件如数值调节按钮、单选按钮和分组框控件，以及列表框、组合框控件和标签控件按钮控件，常用的 ActiveX 控件可以设置控件属性。本章要学习的内容和目标如下所示：

- ❑ 了解 Excel 2010 中的控件类别；
- ❑ 掌握表单控件的使用方法，能够在工作表熟练使用表单控件；
- ❑ 掌握常见的 Excel 2010 中 ActiveX 控件的使用方法，能够熟练地进行控件编程。

14.1　认识 Excel 表单控件

窗体，也称为表单，是可以收集用户输入信息的界面。在 Excel 2010 中，Excel 工作表实际上就是一个窗体。所谓的控件是放置在窗体上的一些图形对象，使用它们能够显示或输入数据、执行某种操作或方便窗体信息的阅读，如，在程序中常见的文本框、列表框、单选按钮和命令按钮等均是控件。控件为用户提供了可供选择的选项，帮助用户实现特定的操作。Excel 2010 工作表中可以使用两类控件，它们是表单控件和 ActiveX 控件。在实际工作中，使用这些控件来构建工作表窗体。

Excel 2010 的表单控件就是早期版本中的窗体控件，其用于控制宏或过程的运行。表单控件在使用时，数据源直接来源于工作表，使用中只需要进行简单的设置而不需要编程就可以实现工作表数据的获取或向工作表中输入数据。在工作表中使用表单控件可以按照下面的步骤进行操作：

（1）在 Excel 2010 功能区中选择“开发工具”选项卡，单击“控件”组的“插入”按钮，在打开的列表的“表单控件”栏中选择需要的控件。在工作表中拖动光标即可创建该控件，如图 14.1 所示。

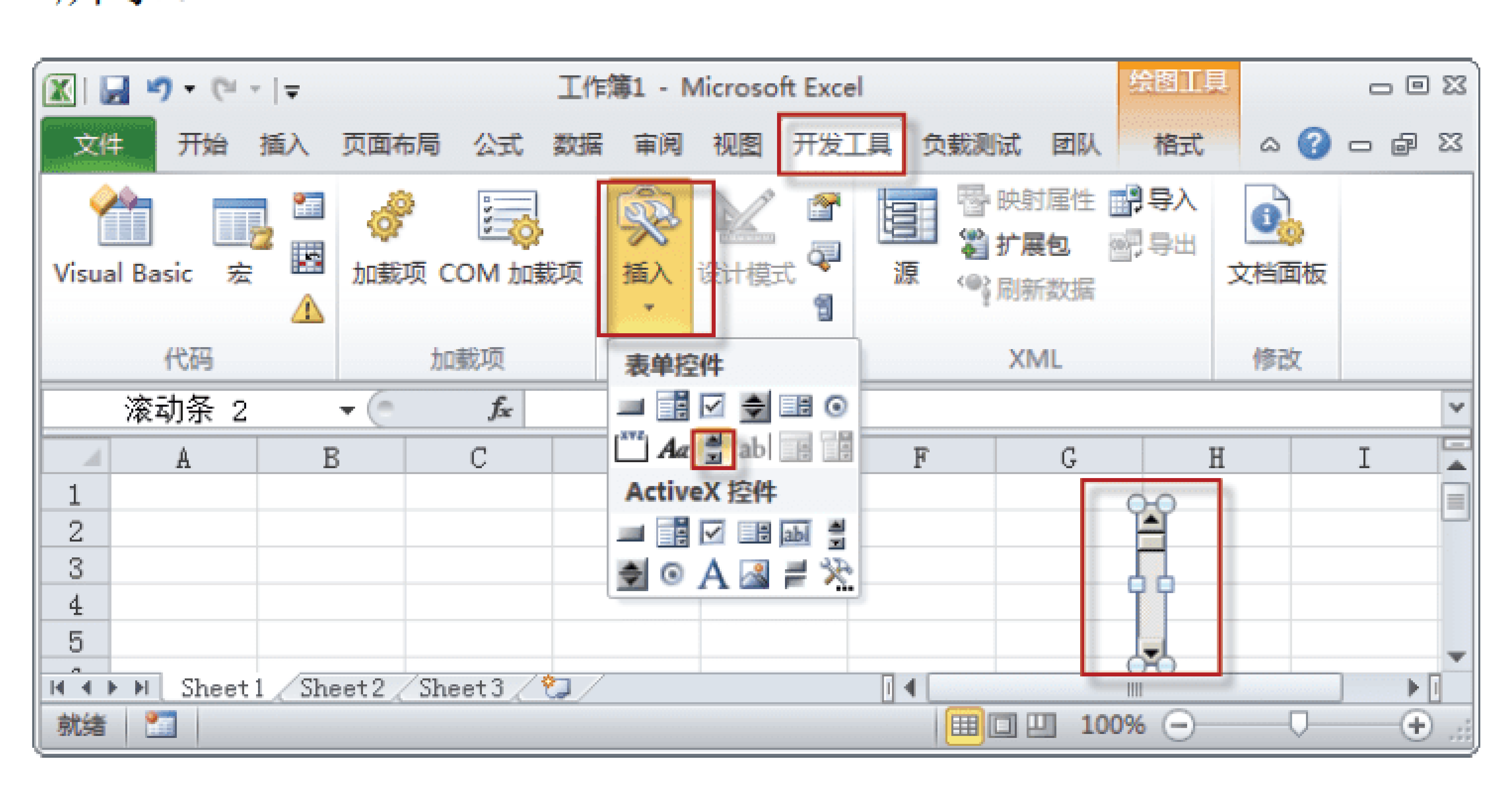


图 14.1　创建控件

🔔**提示：**在“表单控件”列表中一共有 12 个控件，但只有前面的 9 个控件能够在工作表中使用，后面的 3 个控件只能在窗体中使用。

（2）在添加控件后，控件的大小和位置可以改变。选择控件，拖动控件边框上的 8 个控制柄，可以改变控件的大小，如图 14.2 所示。将光标放置于控件中，拖动控件可以改变控件在工作表中的位置，如图 14.3 所示。

（3）在创建控件后，Excel 会为控件指定一个名字，在过程中可以使用这个名字来实现对控件的引用。这个名字在实际使用中是可以更改的。选择控件，在 Excel 2010 工具栏左侧的名称框中输入需要的控件名称后按 Enter 键，即可对控件更名。

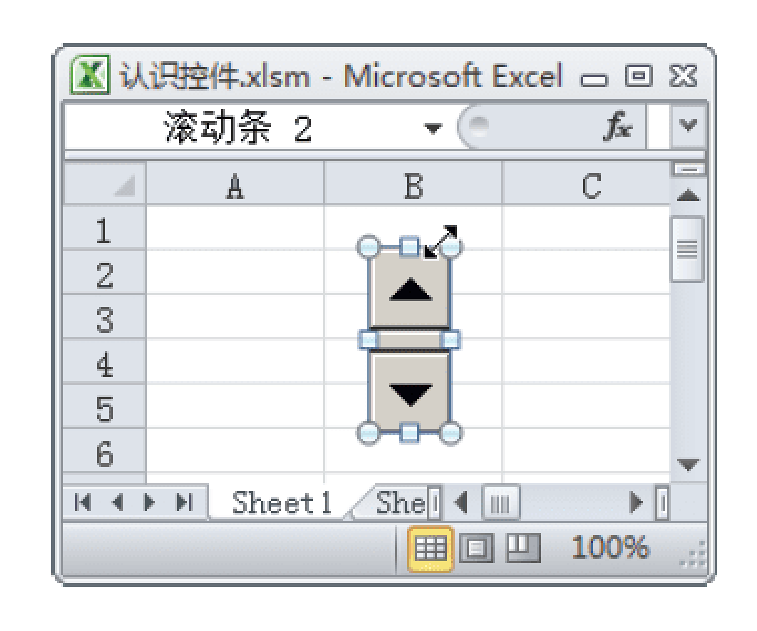



图 14.2　改变控件的大小

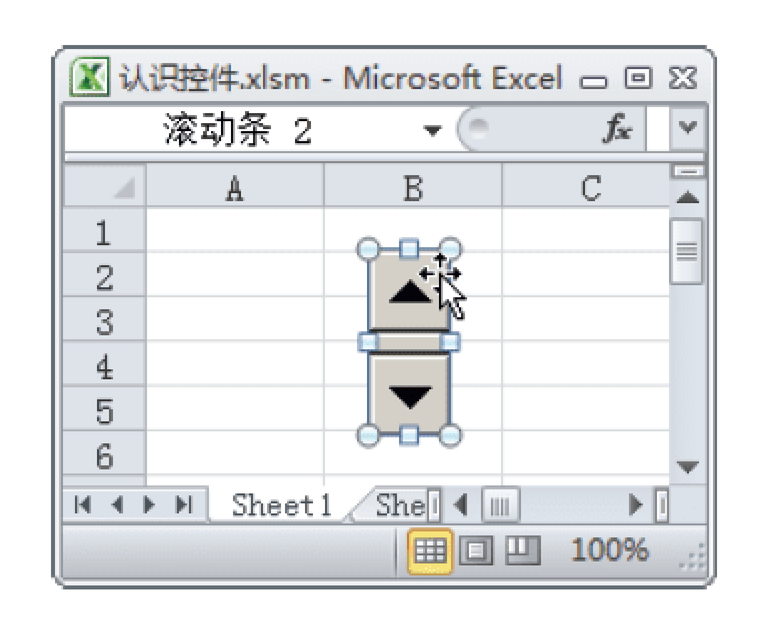


图 14.3　移动控件的位置

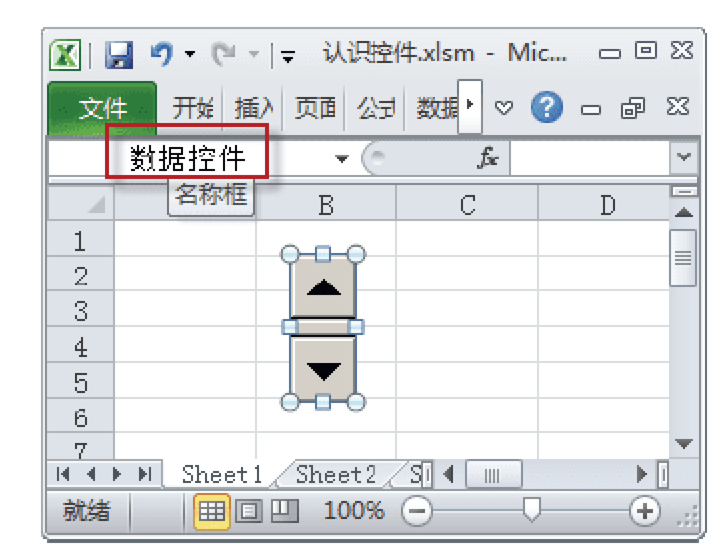


图 14.4　控件更名

🔔**提示：**在对控件进行操作前，必须首先选择该控件。控件不同，选择的方法也略有不同。大多数控件可以单击将其选中。而部分控件如“选项按钮”和“复选框”等需要右击才能选中，因为单击会改变控件的状态。

14.2　使用 Excel 表单控件

表单控件实际上是一些简单控件，与用户窗体中使用的控件相比，其不具有属性，也没有事件过程，但可以执行宏。本节将介绍一个用户意见反馈系统的制作过程，通过这个实例的制作，来逐个介绍表单控件的使用方法。

14.2.1　“数值调节按钮”控件

“数值调节按钮”控件可用来输入指定范围内的一个数值。在工作表中，单击控件的向上箭头将增大数值，单击向下箭头将能减小数值。该控件的功能类似于“滚动条”控件，但其没有中间的滑块。下面介绍在工具表中使用该控件的方法。

（1）打开工作表，单击“开发工具”选项卡的“插入”按钮，在打开列表中选择“数值调节按钮”控件，在工作表中拖动光标绘制控件，将控件插入到工作表中，如图 14.5 所示。

（2）右击控件，在弹出的快捷菜单中选择“设置对象格式”命令，打开“设置对象格式”对话框。选择“控制”选项卡，对各设置项进行设置，如图 14.6 所示。

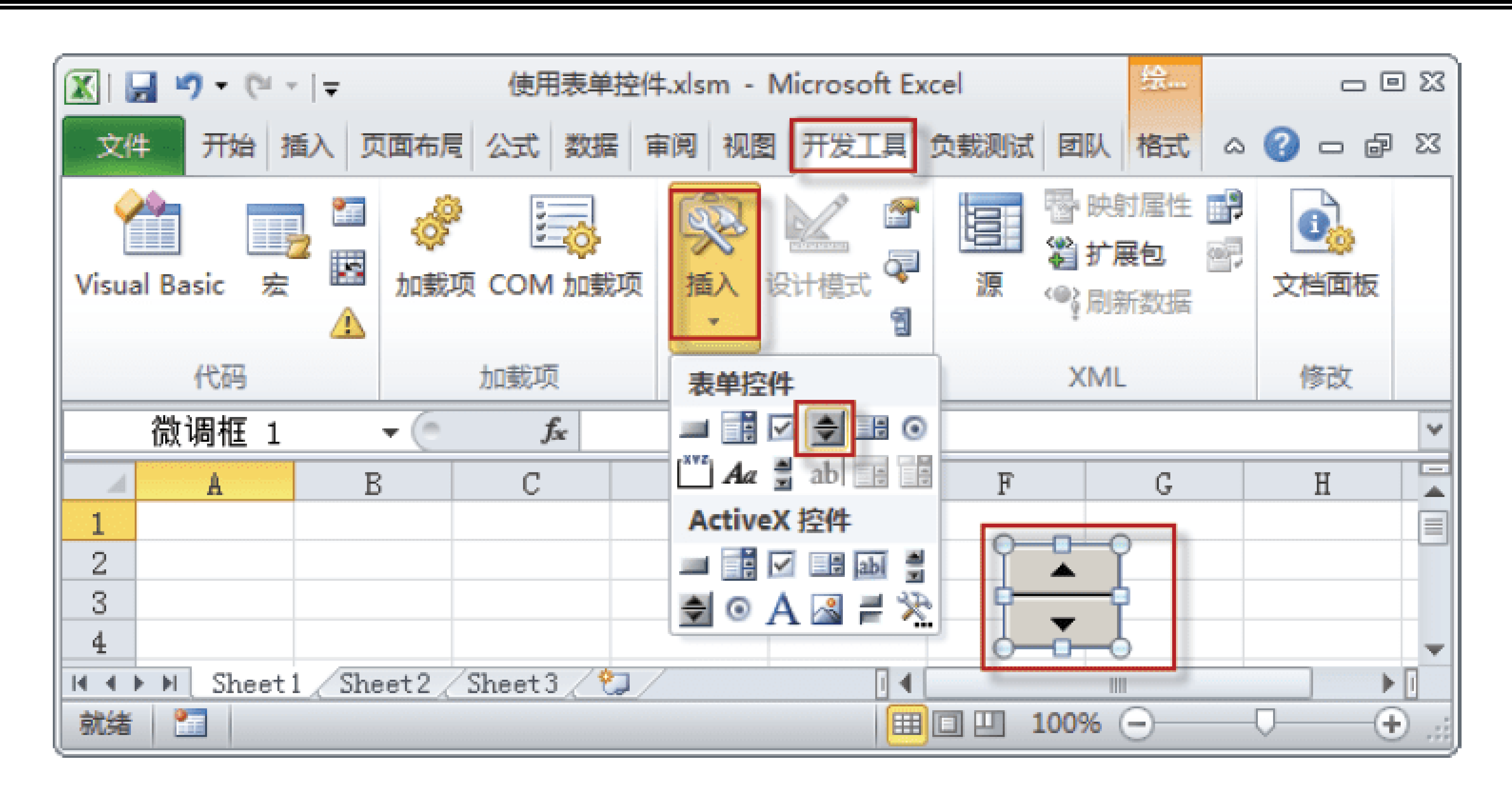


图 14.5　添加“数值调节按钮”控件



图 14.6　“设置控件格式”对话框中的设置

提示：“数值调节按钮”控件用来调整数值的大小。图 14.6 中的“最小值”和“最大值”表示可调节的数字的最小和最大的值，而“步长”表示每次单击按钮数值改变的大小。“单元格链接”用于设置和控件关联的单元格，控件的“当前值”在这个关联的单元格中显示，改变单元格中的数值，控件的“当前值”也会发生改变。

（3）取消控件的选择，单击控件上的按钮，指定单元格的数据发生变化。每次增大或减小设定的“步长”值，如图 14.7 所示。

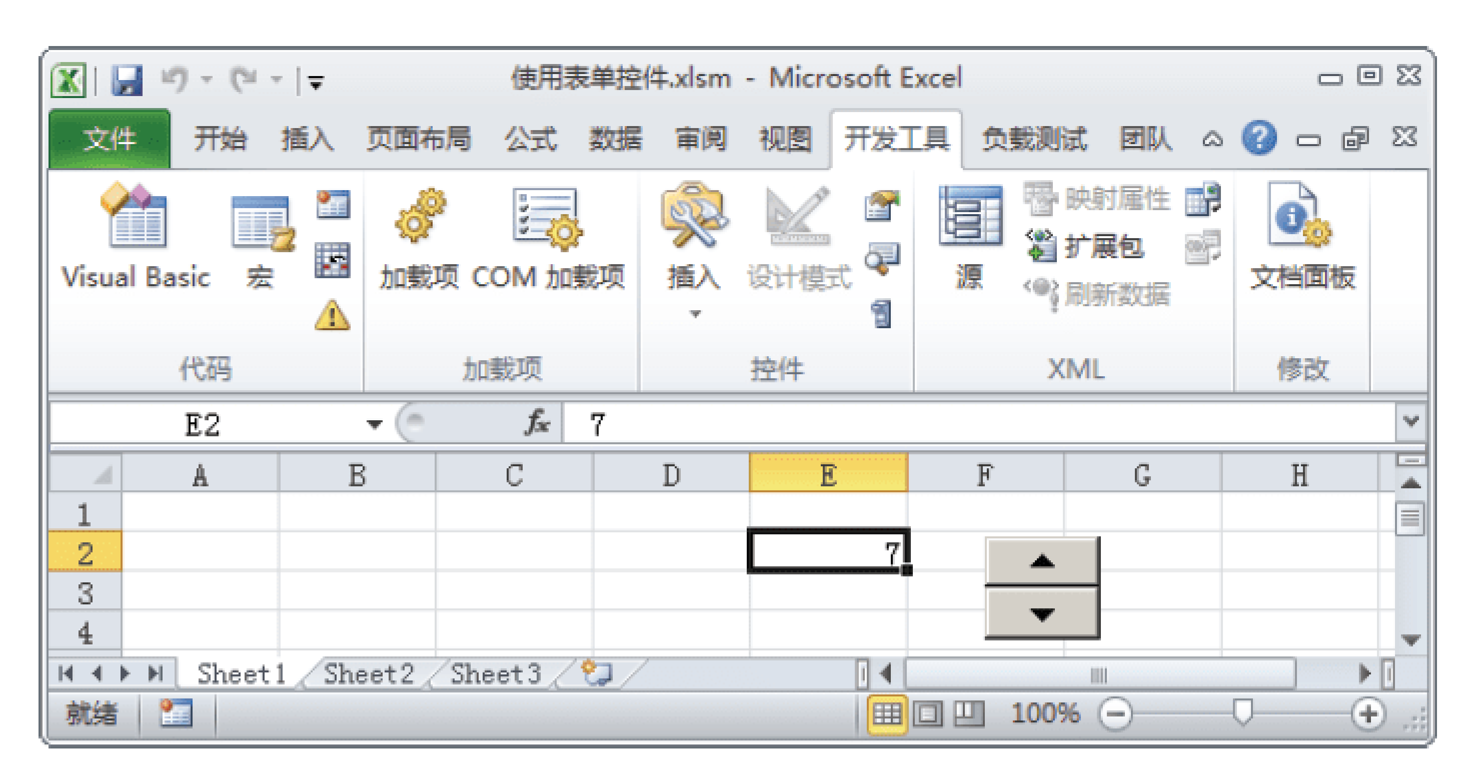


图 14.7　“数值调节按钮”使用效果

14.2.2 “单选按钮”控件和“分组框”控件

“单选按钮”控件一般成组出现，当组中的一个“单选按钮”处于选中状态时，其他按钮将自动处于未被选择状态。“单选按钮”控件常常与“分组框”控件一起使用，使用“分组框”控件来对其进行分组，同组的“单选按钮”只允许一个被选择。下面介绍“分组框”控件和“单选按钮”控件配合使用的方法。

（1）在工作表中添加一个“分组框”控件，如图 14.8 所示。在“分组框”控件的标题栏上单击，将标题改为“性别”，如图 14.9 所示。

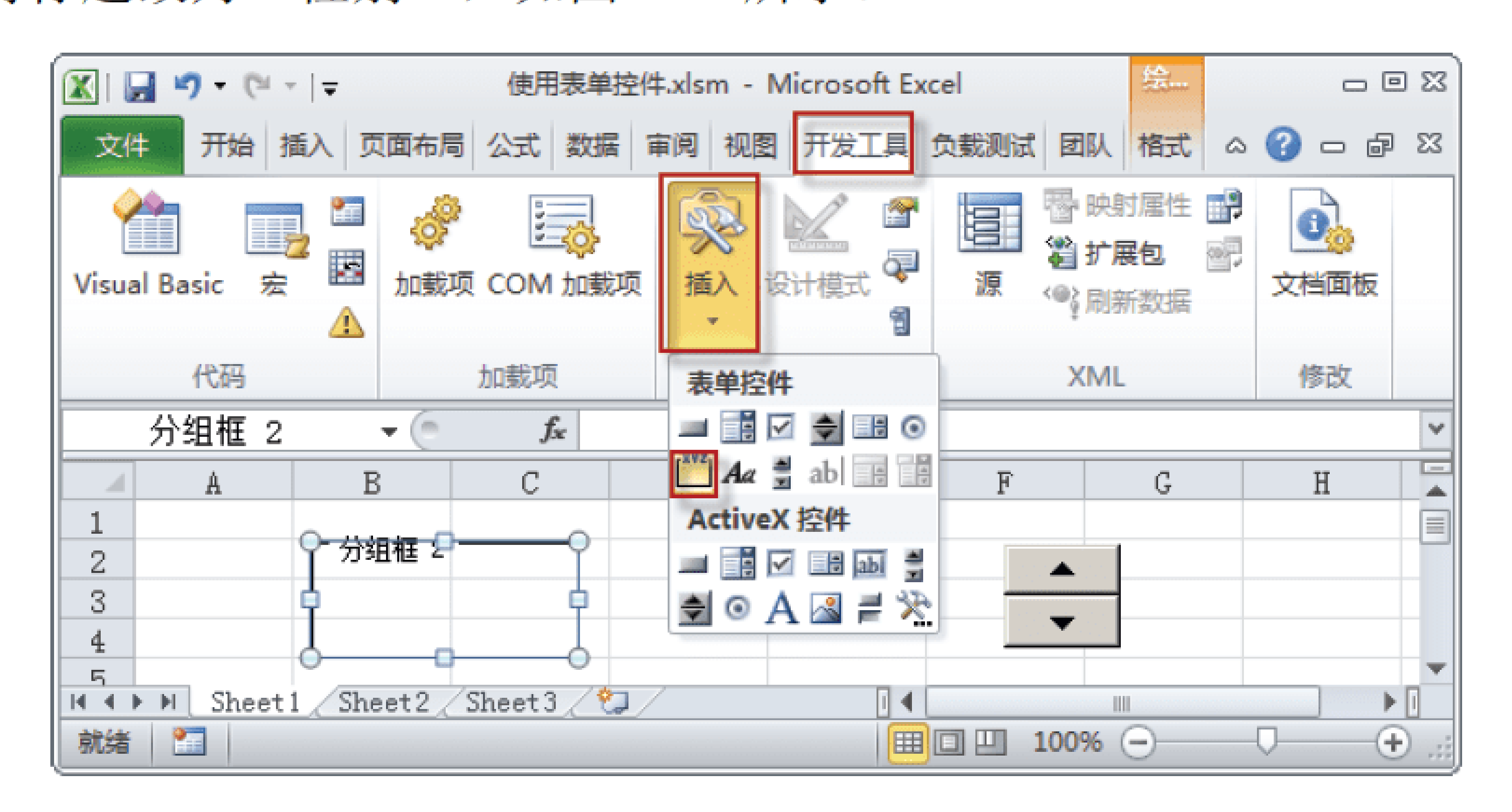


图 14.8　添加一个分组框

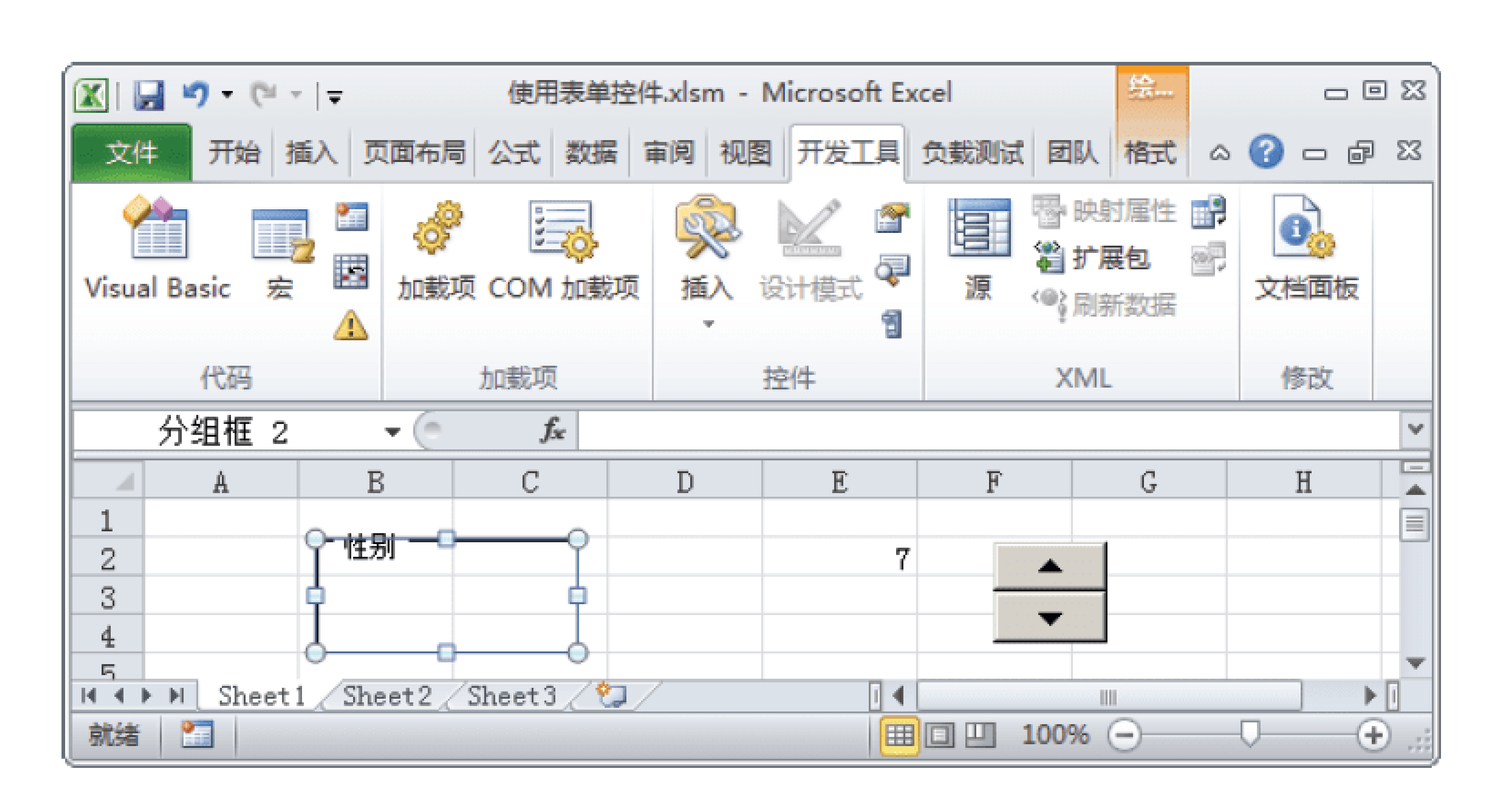


图 14.9　更改标题

（2）向分组框中添加 2 个“单选按钮”控件，如图 14.10 所示。分别在“单选按钮”控件上右击选择按钮，更改按钮的标题，如图 14.11 所示。

（3）右击“单选按钮”，在弹出的快捷菜单中的选择“设置控件格式”命令，打开“设置控件格式”对话框。在“控制”选项卡的“单元格链接”栏中输入与控件链接的单元格地址，如图 14.12 所示。

提示： 被“分组框”控件分为一组的多个“单选按钮”控件，只需要设置一个控件的“单元格链接”即可，另一个会自动链接到这个单元格。

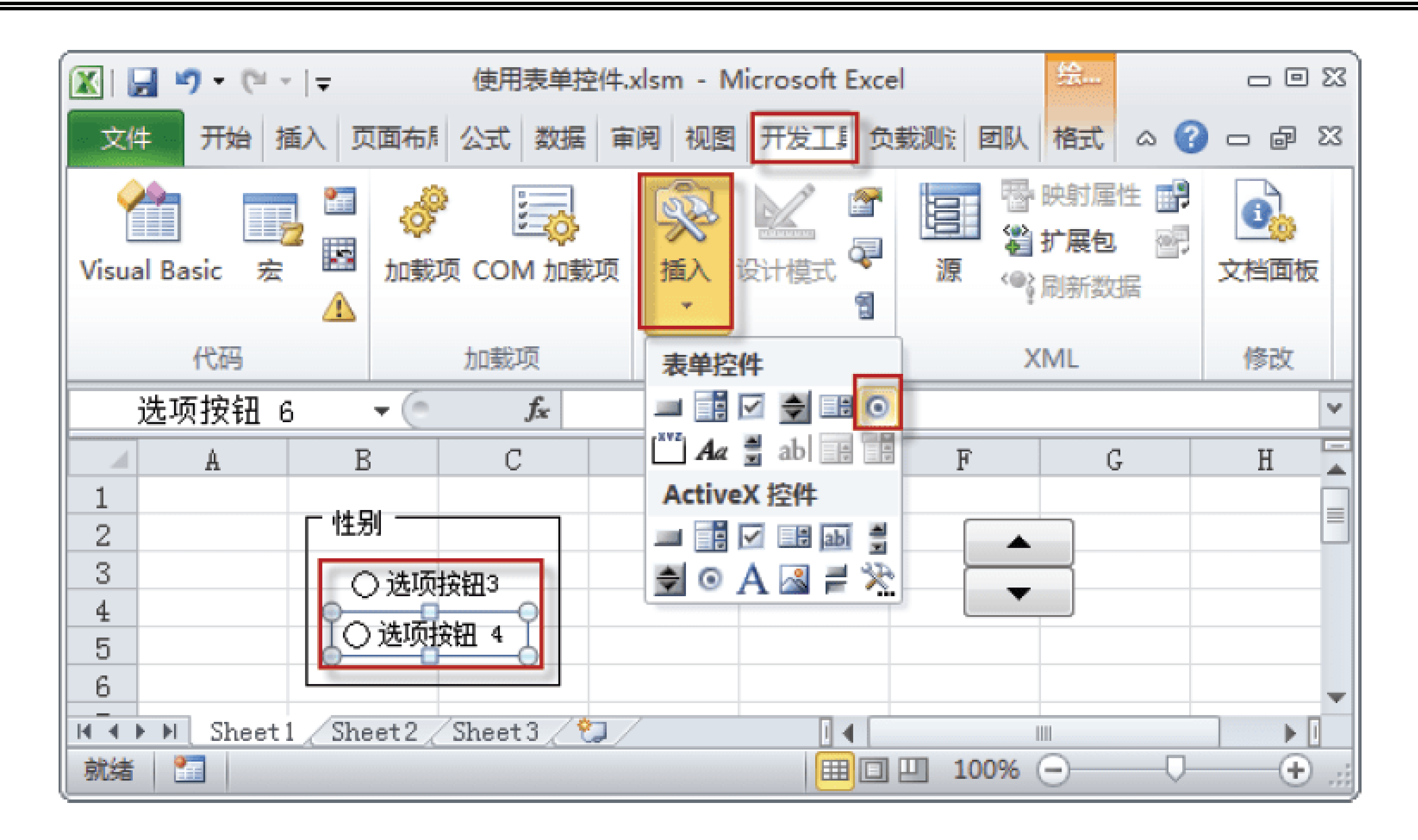


图 14.10　向分组框中添加“单选按钮”控件

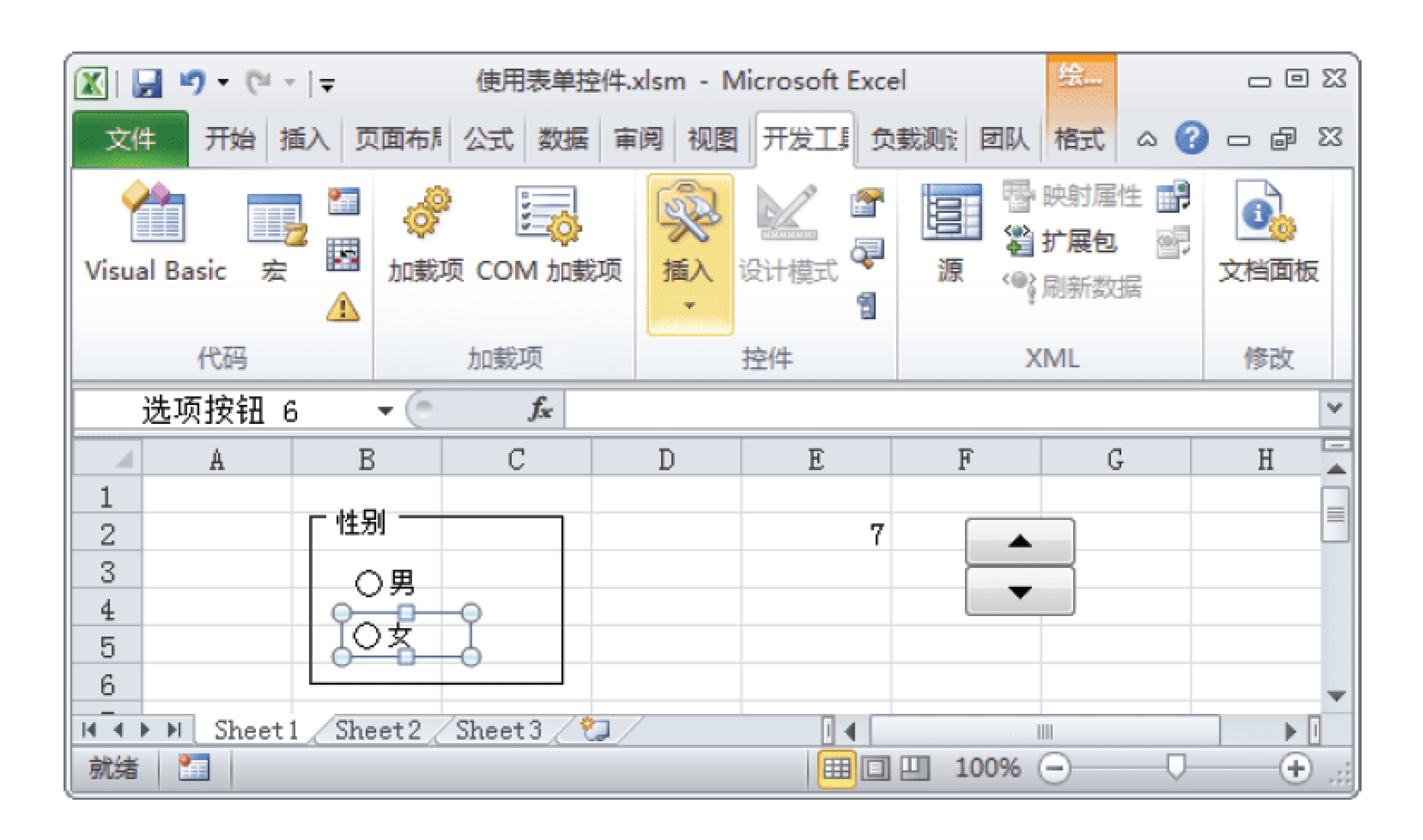


图 14.11　更改控件标题

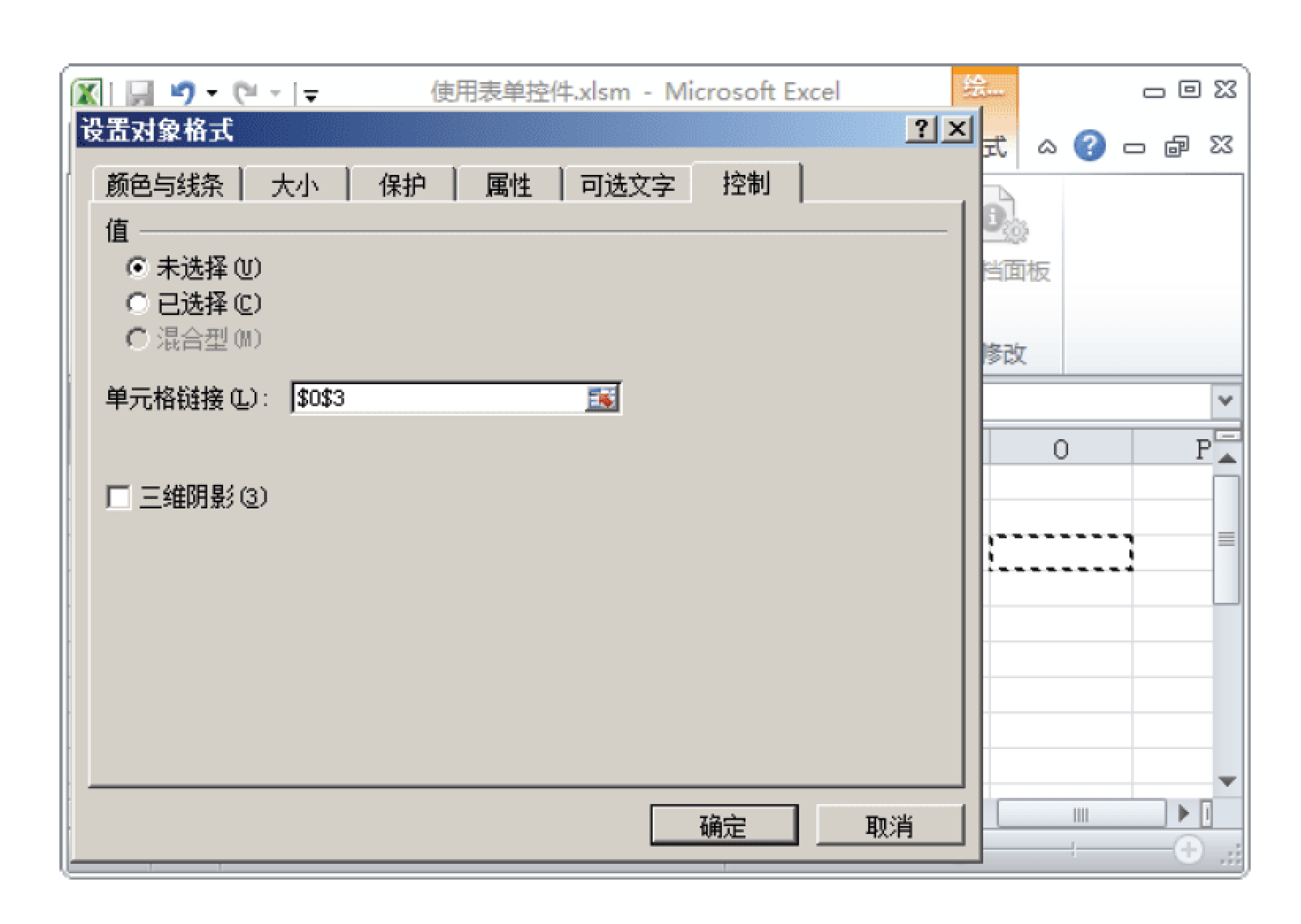


图 14.12　“设置对象格式”对话框的设置

（4）此时，如果需要在工作表中显示单选按钮选择的状态，可以采用下面方法：在另一个单元格中输入公式“=if(O3=1,"男","女")”对单元格数据进行判断即可。此时，单元格将能够根据控件的选择来显示正确的性别，如图 14.13 所示。

（5）使用相同的方法添加第二组“分组框”控件和“单选按钮”控件，并对控件进行类似的设置。此时工作表中控件的布局如图 14.14 所示。

注意：为了美观，两个“分组框”控件的大小必须一样，此时可以打开“性别”分组框的“设置对象格式”对话框，在“大小”选项卡中读出其“高度”和“宽度”的值。然后在“婚否”分组框的“设置对象格式”对话框的“大小”选项卡中使用相同的值即可。

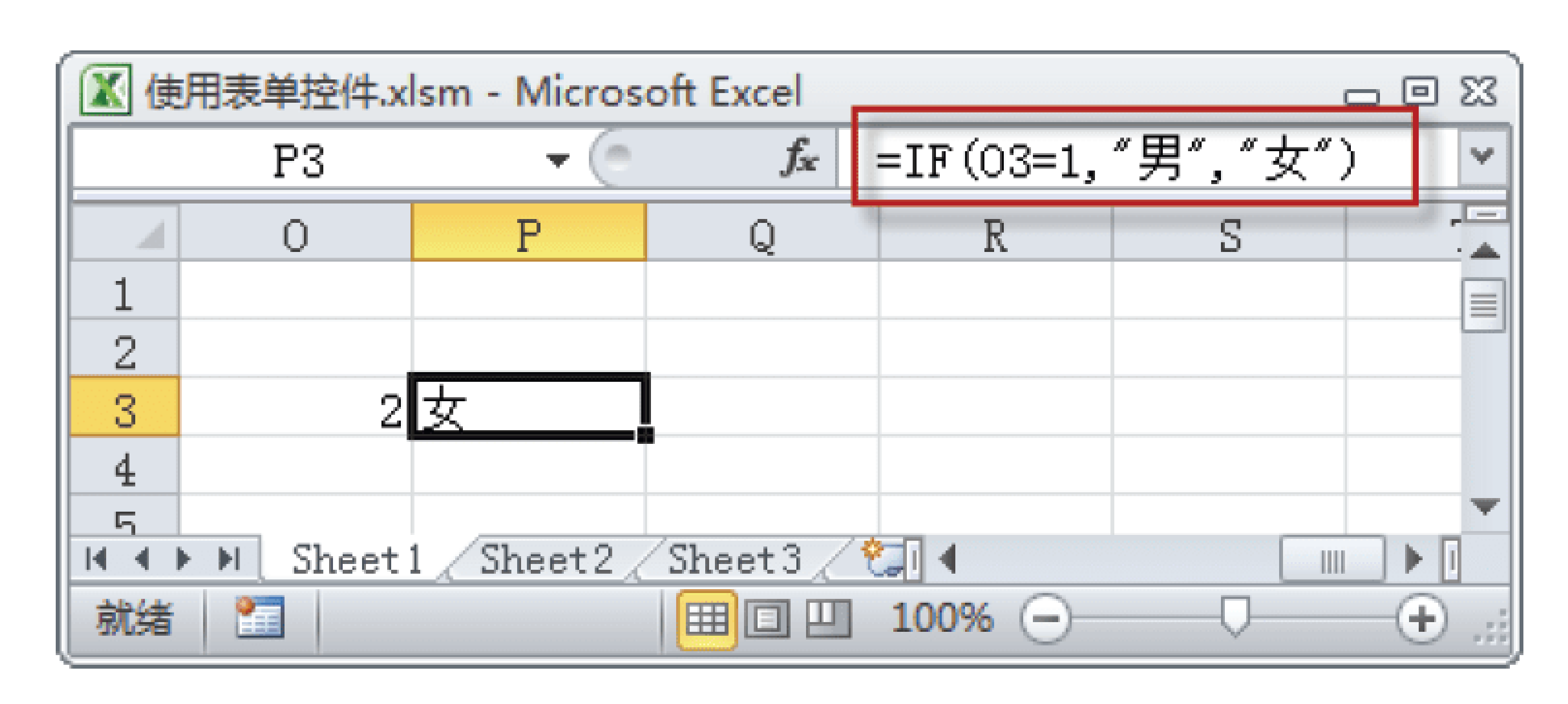


图 14.13　设置单元格公式

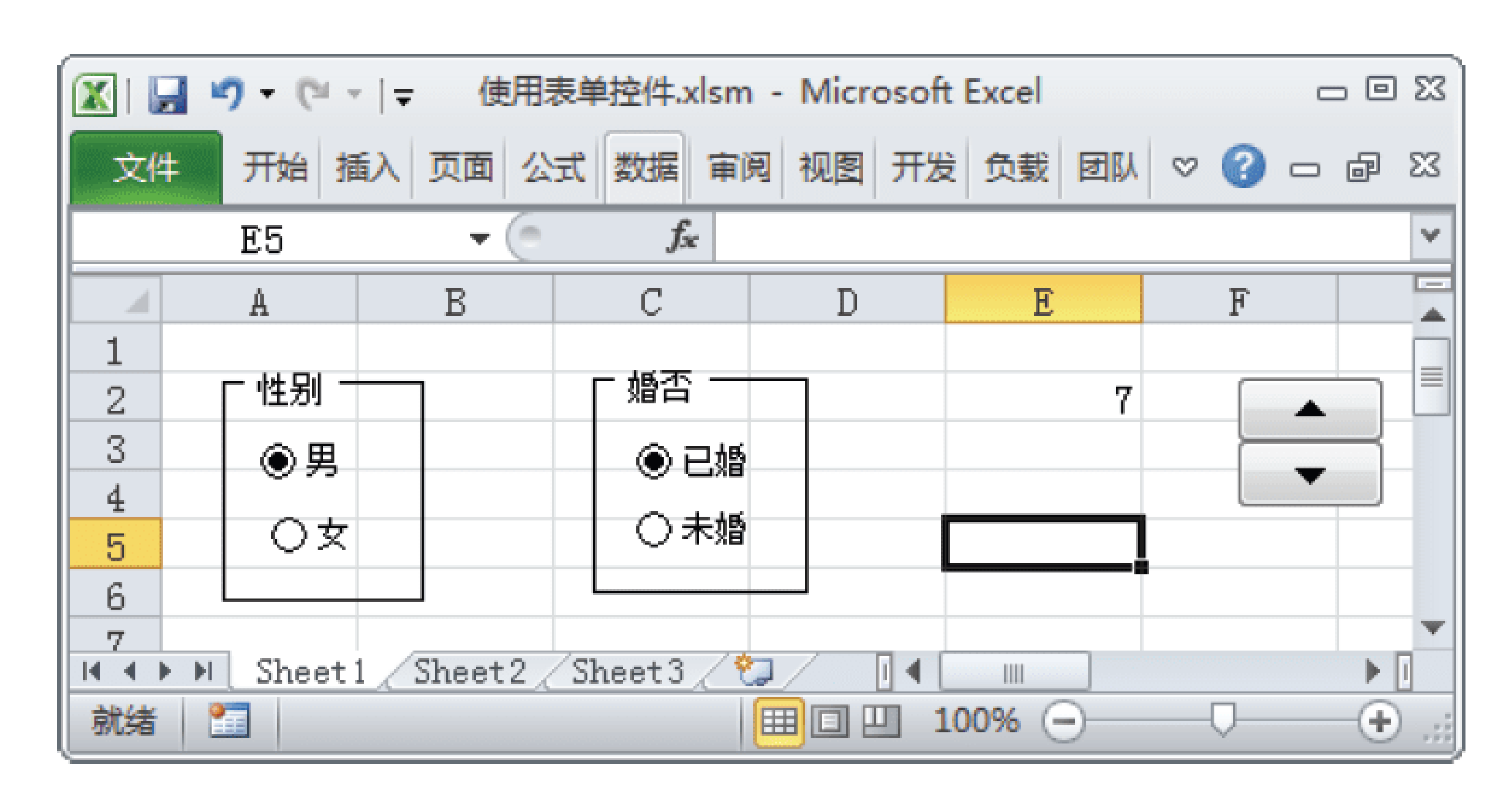


图 14.14　工作表中控件的布局

14.2.3 “组合框”控件

组合框是一种下拉列表框，用户可以通过对下拉列表中选项的选择来实现不同的操作。该控件是窗体中比较常用的一种控件，常用来显示多个数据项，从而为用户提供多个选择。下面介绍“组合框”控件的使用方法。

（1）在工作表中添加一个“组合框”控件，如图 14.15 所示。在工作表的 O8 至 O16 单元格中输入组合框各个选项的内容。选择该控件，打开“设置控件格式”对话框，再选择“控制”选项卡。在“数据源区域”输入框中输入选项所在的单元格地址。在“单元格

链接”文本框中输入选择结果存储的单元格，如图 14.16 所示。

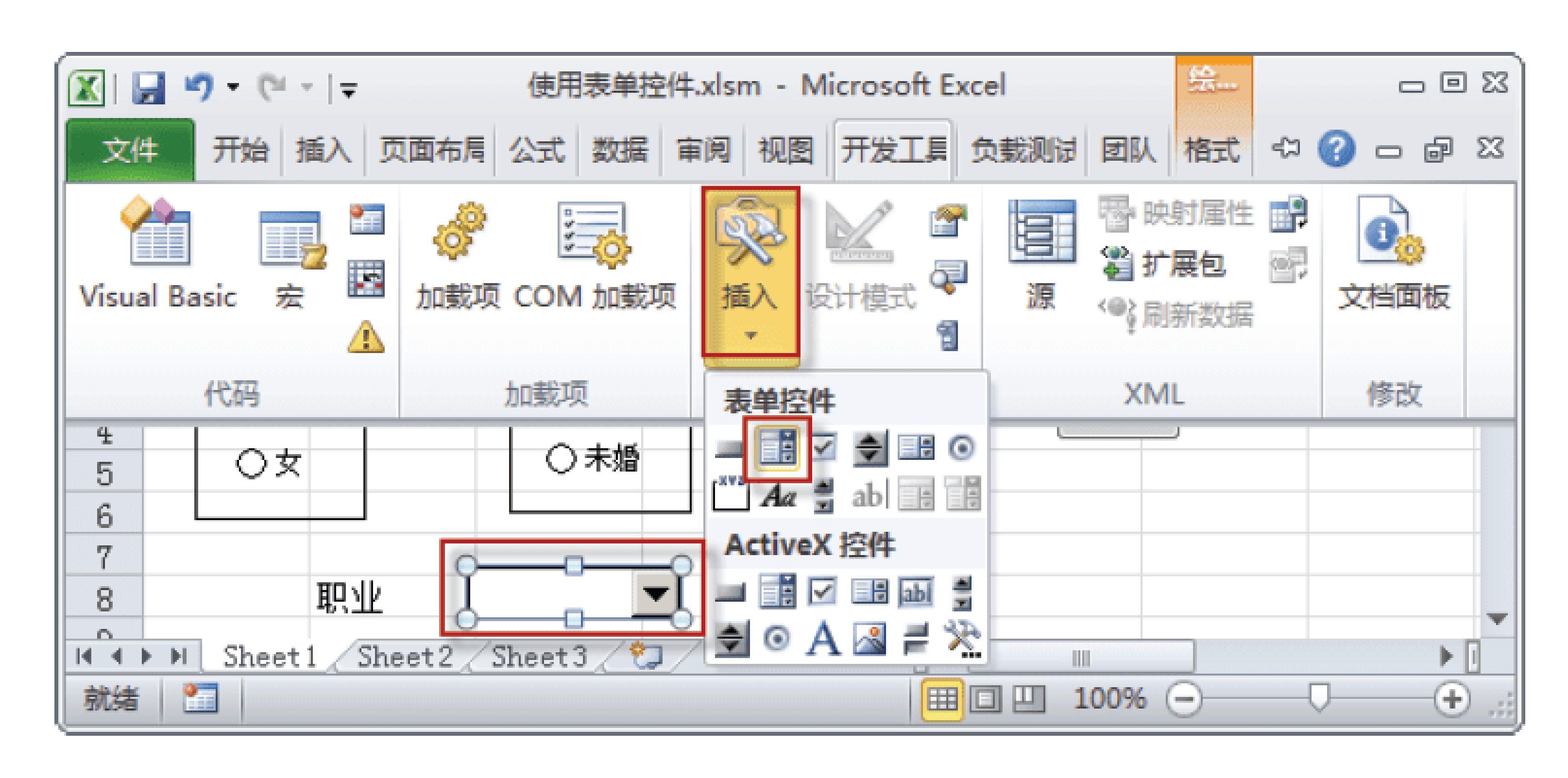


图 14.15　添加“组合框”控件

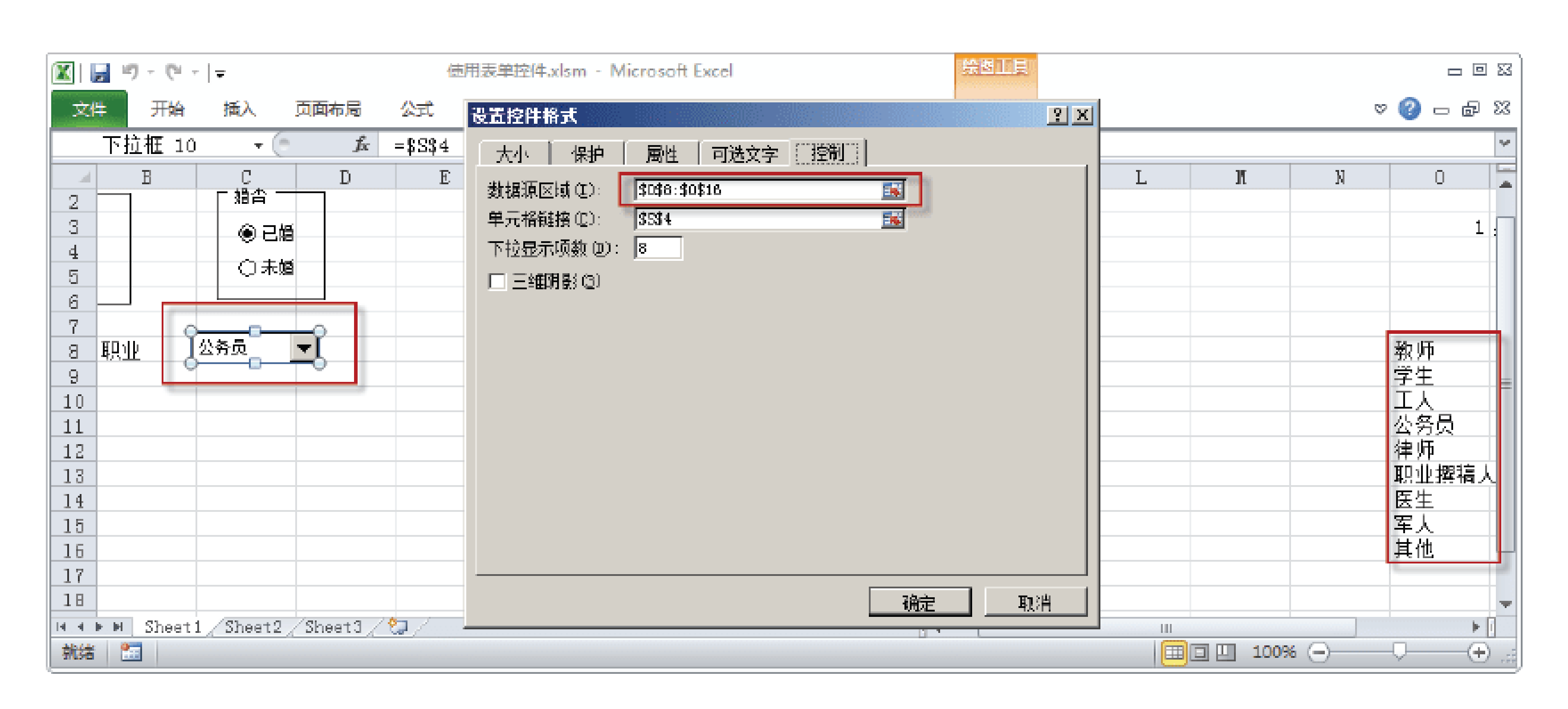


图 14.16　“控制”选项卡的设置

提示：在“控制”选项卡中，“下拉显示项数”文本框中的数值指定组合框下拉列表框中一次可以显示的最大项数。

（3）在工作表的 T1 单元格中输入公式“=INDEX(O8:O16,S3)”，此单元格中将显示“组合框”控件的选项，如图 14.17 所示。

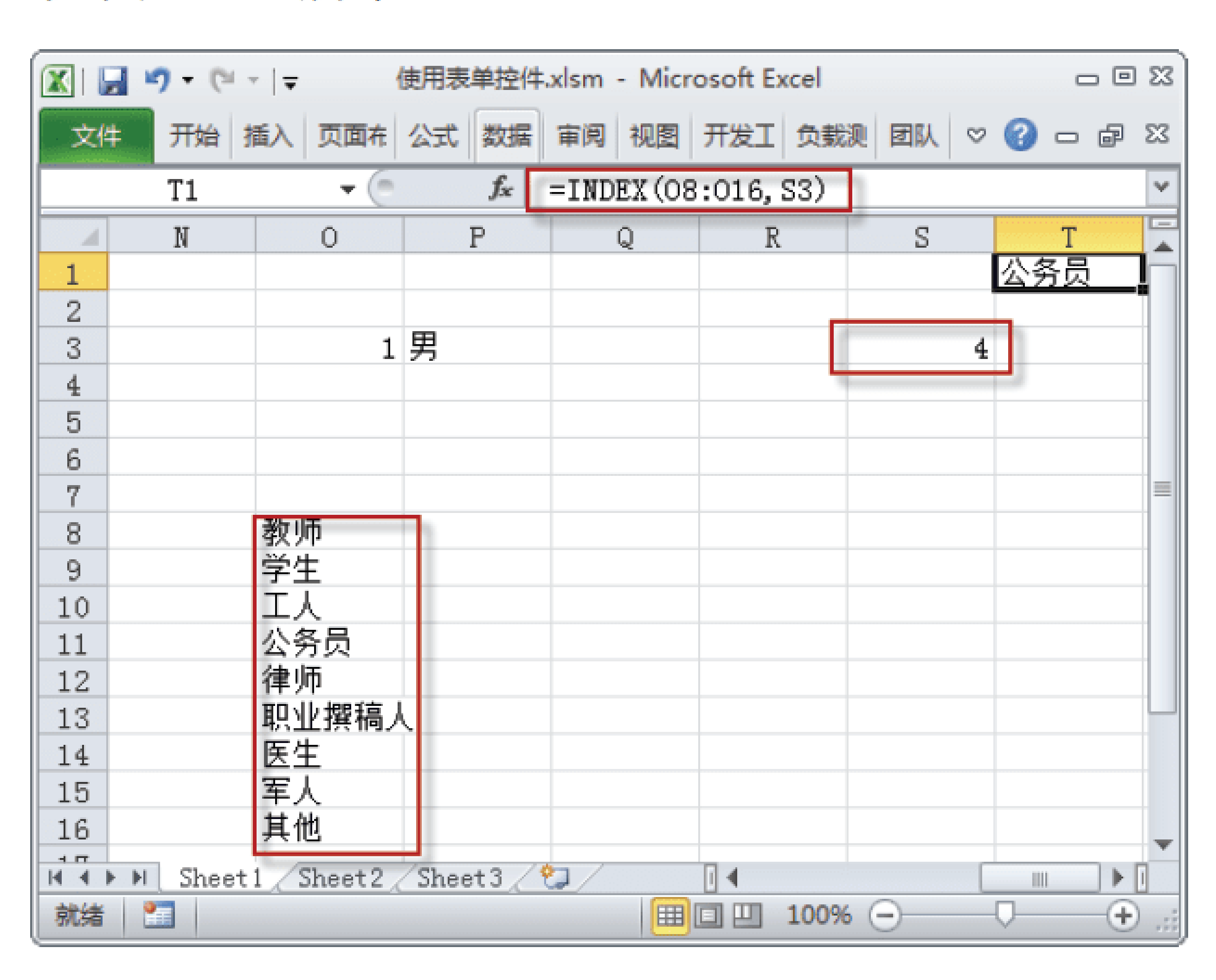


图 14.17　显示“组合框”控件中的选项

提示：这里，Index 函数将返回对指定区域的值的引用，前面指的是 O8:O16 是选项单元格区域。S3 指的是“组合框”控件中选项的编号。

（4）使用上面介绍的相同步骤，在工作表中添加第二个“组合框”控件，并对控件进行设置。此时工作表的布局如图 14.18 所示。

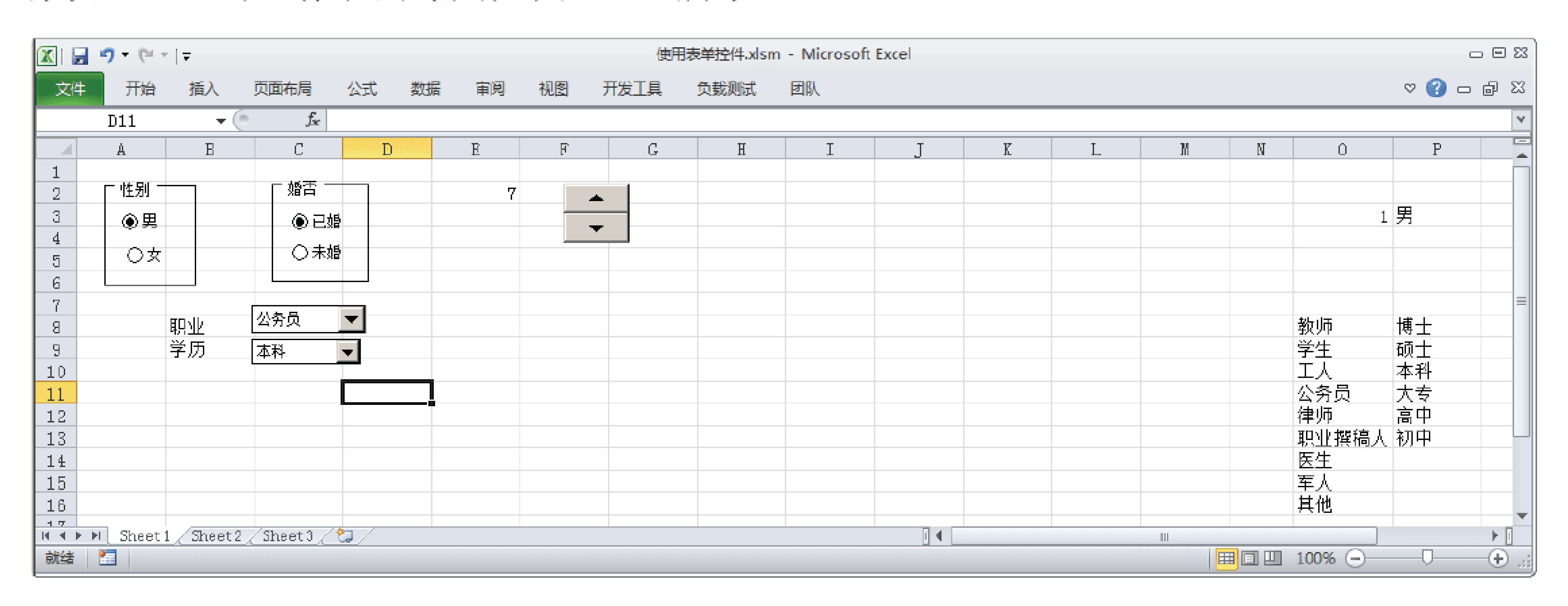


图 14.18　工作表的布局

14.2.4 “列表框”控件

“列表框”控件也用于在多个项目中进行选择，但与“组合框”控件不同的是，其可以显示多个项目，而不是像“组合框”控件那样只显示一个。如果项目很多，控件自动在右侧添加垂直滚动条。另外，“列表框”控件可以设置选择多个选项。

（1）向工作表中添加一个“列表框”控件，如图 14.19 所示。在工作表的 Q8 至 Q13 单元格中输入“列表框”控件各个选项内容。选择“列表框”控件打开控件的“设置控件格式”对话框的“控制”选项卡，将“数据源区域”指定为刚才各选项所在的单元格，如图 14.20 所示。

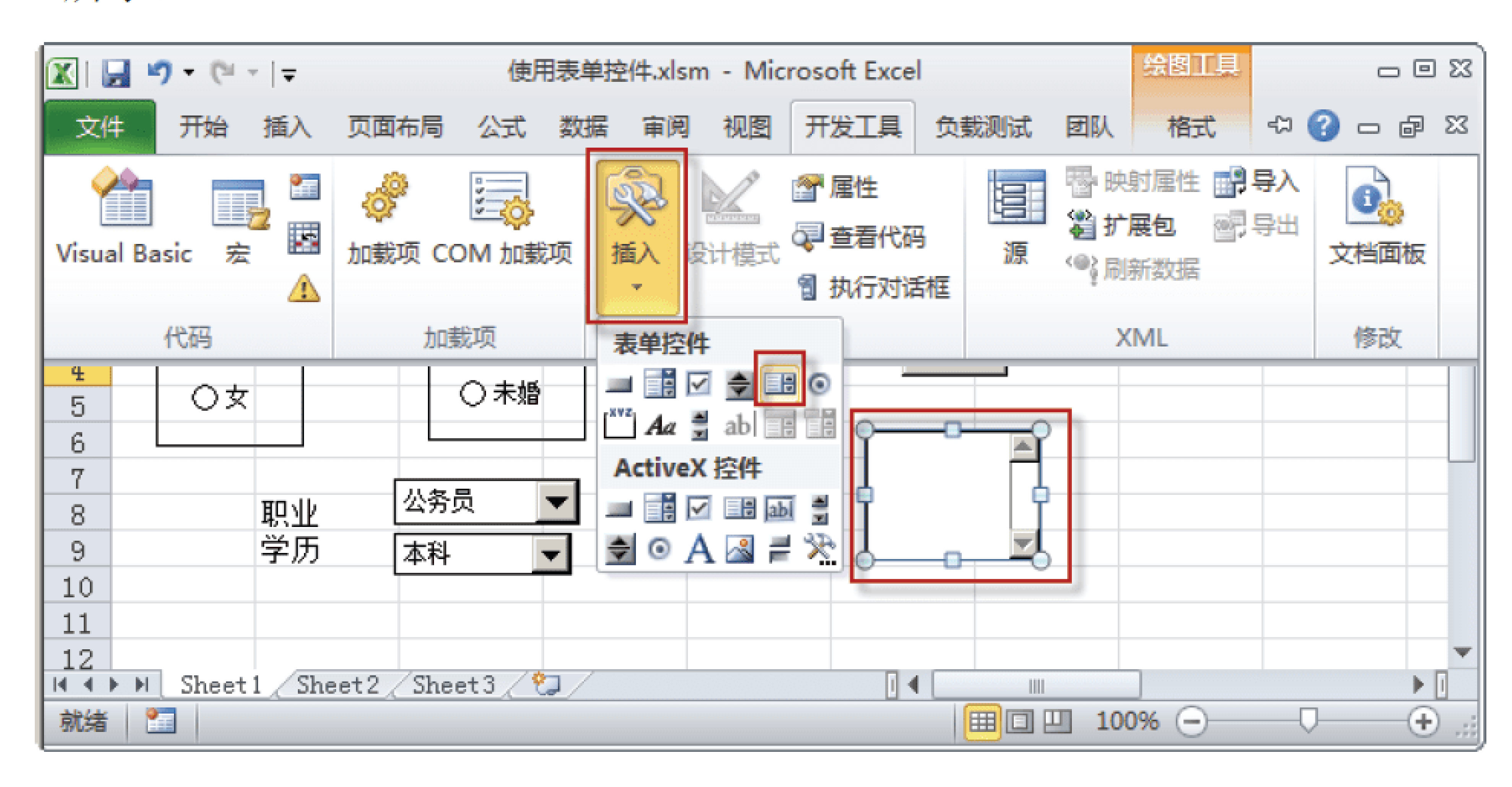


图 14.19　添加“列表框”控件

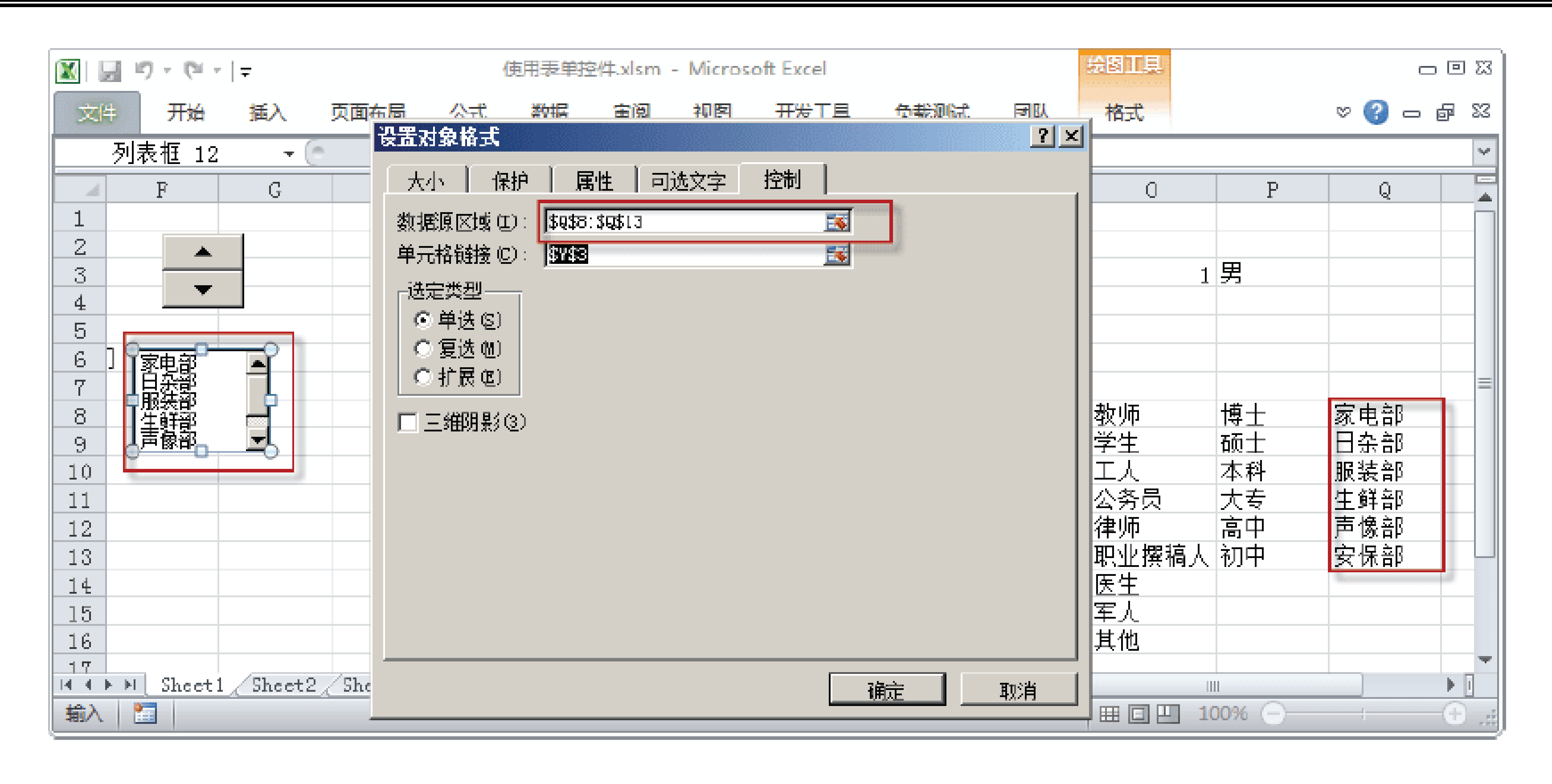


图 14.20　“控制”选项卡的设置

提示： 与“组合框”控件不同的是，“列表框”控件在“控制”选项卡中“选定类型”组中可以设置同时选择的选项的多少。其中，“单选”表示只能选择一项；“复选”表示能够通过单击的方式选择多个选项；“扩展”表示可以选择多个选项，但需要按住 Ctrl 键，然后单击鼠标来进行选择。

（2）关闭“设置控件格式”对话框，使用与“组合框”控件相同的设置方法将选项填入指定单元格中，如图 14.21 所示。

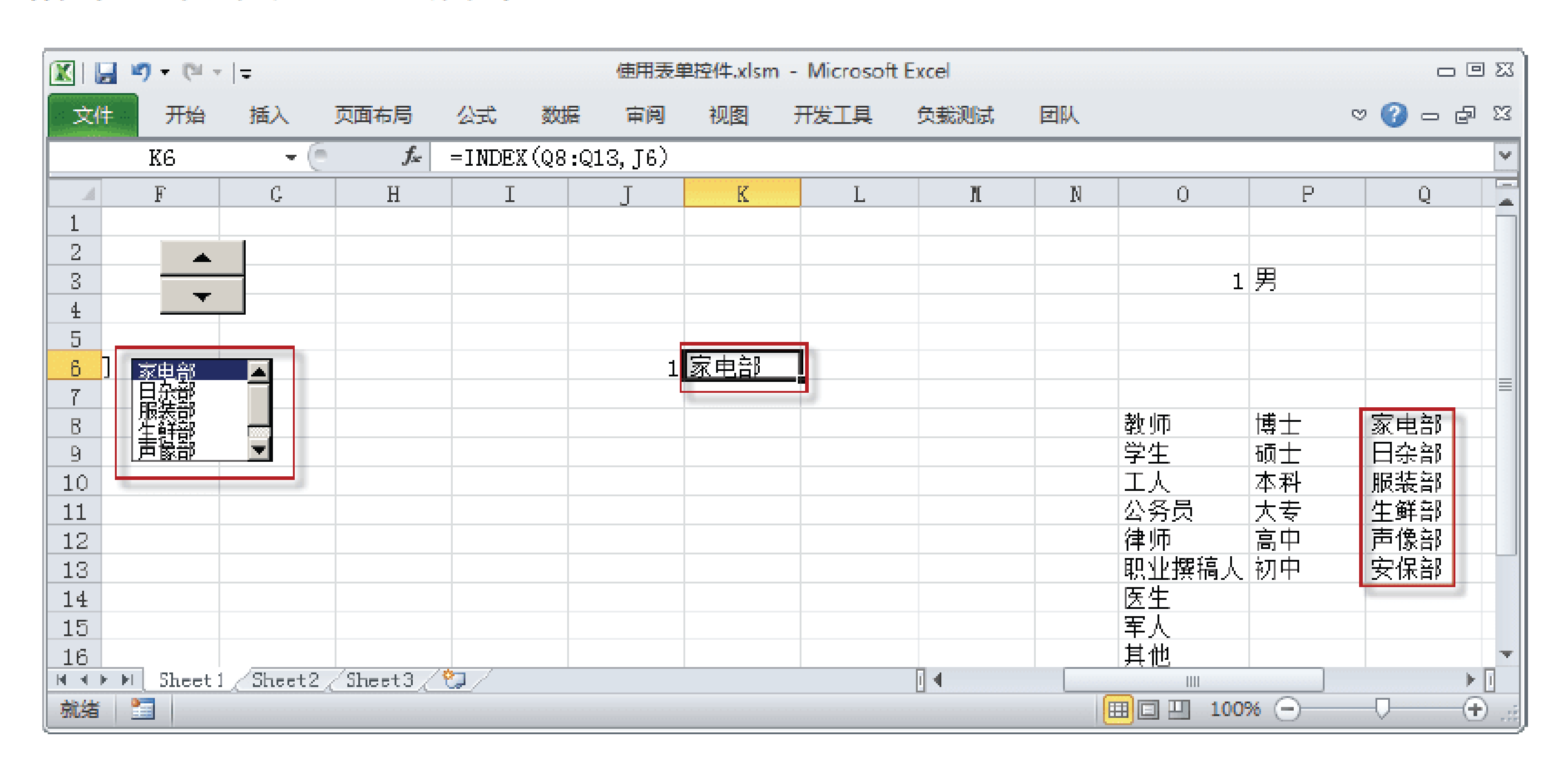


图 14.21　将选项填入单元格

14.2.5 “复选框”控件

在工作表窗口中，经常需要用户直接进行选择需要的项目，此时可以使用“复选框”控件来实现。与“单选按钮”相比，多个“复选框”控件可以同时选择，也就是能够实现多选。同时，在选择了“复选框”控件后，控件会显示一个“√”，此时再次单击将取消选中“复选框”控件。这意味着单击该控件可以在选择与取消间进行切换。

（1）在工作表中添加一个“复选框”控件，如图 14.22 所示。使用与“单选按钮”相同的方法设置按钮的标题。修改控件标题，打开“设置控件格式”对话框对控件进行设置，如图 14.23 所示。

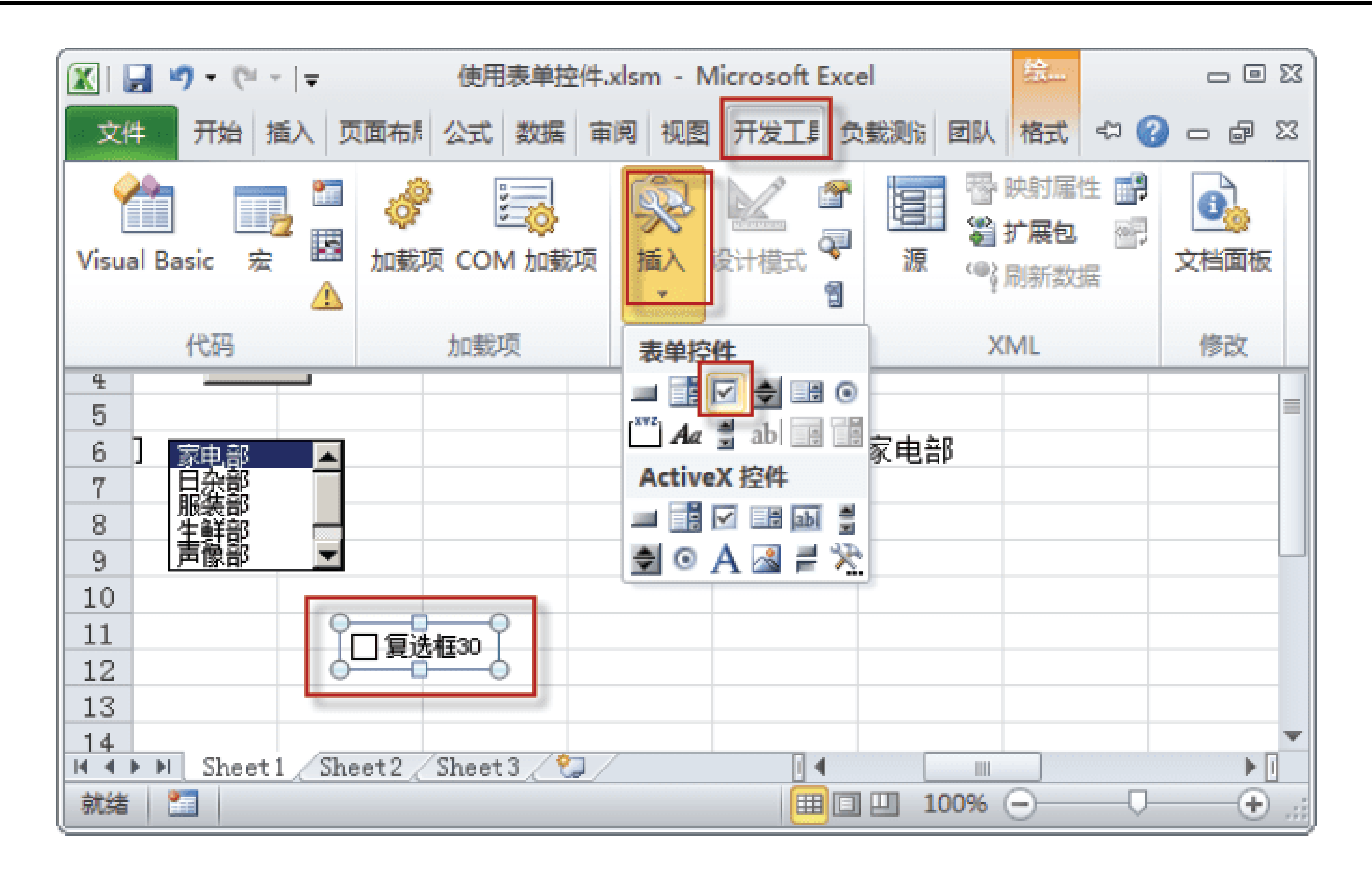


图 14.22　添加“复选框”控件

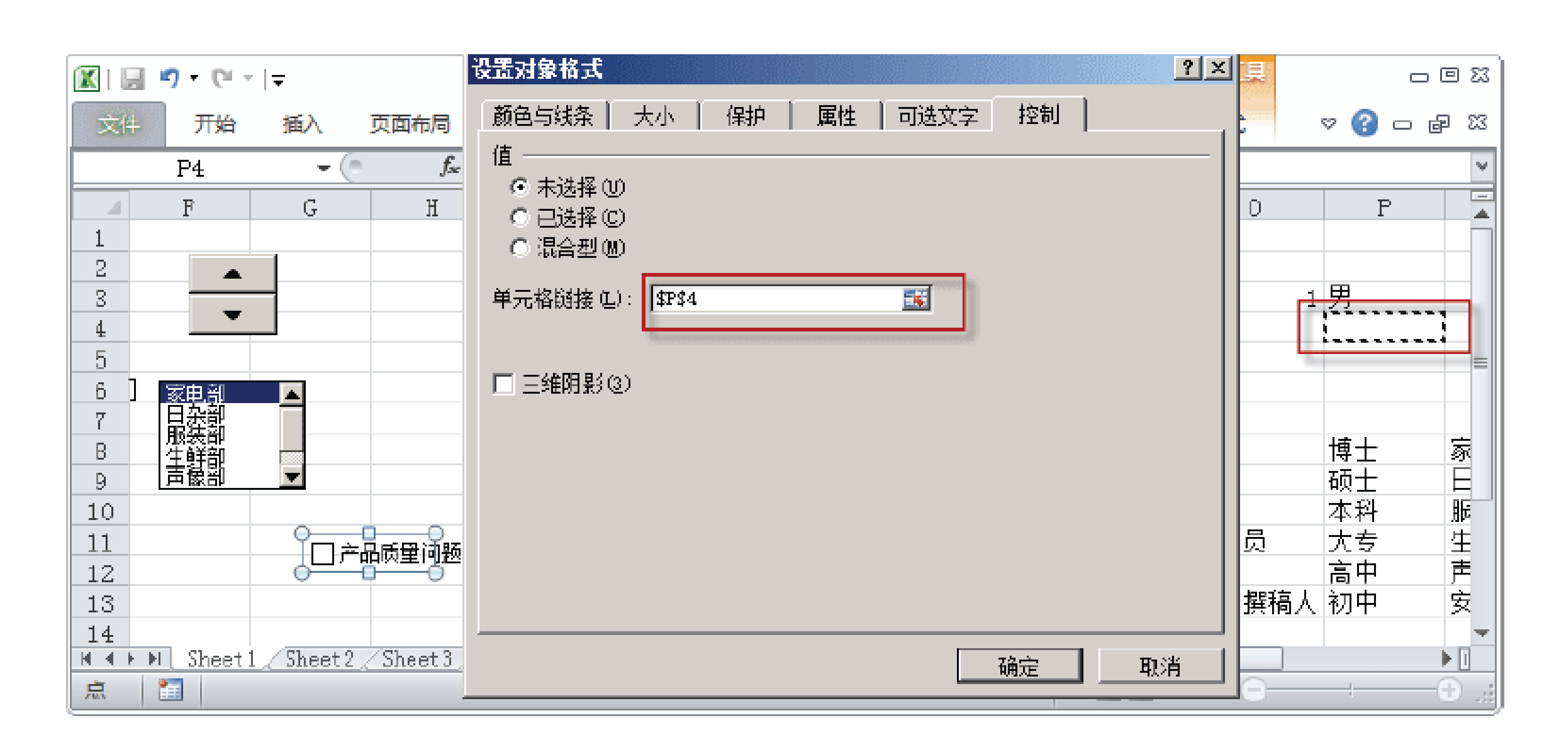


图 14.23　设置“单元格链接”

（2）在工作表中再添加 3 个“复选框”控件，分别修改其标题，设置链接单元格。完成设置后，在指定的单元格中显示控件的选择状态，如图 14.24 所示。

提示： 在“设置对象格式”对话框的“控制”选项卡中，如果选中“未选择”单选按钮，表示控件的初始状态是没有选择，此时链接单元格显示“False”。如果选中“已选择”单选按钮，表示控件的初始状态是已经选择，链接单元格显示“True”。如果选中“混合型”单选按钮，则链接单元格显示“#N/A”。

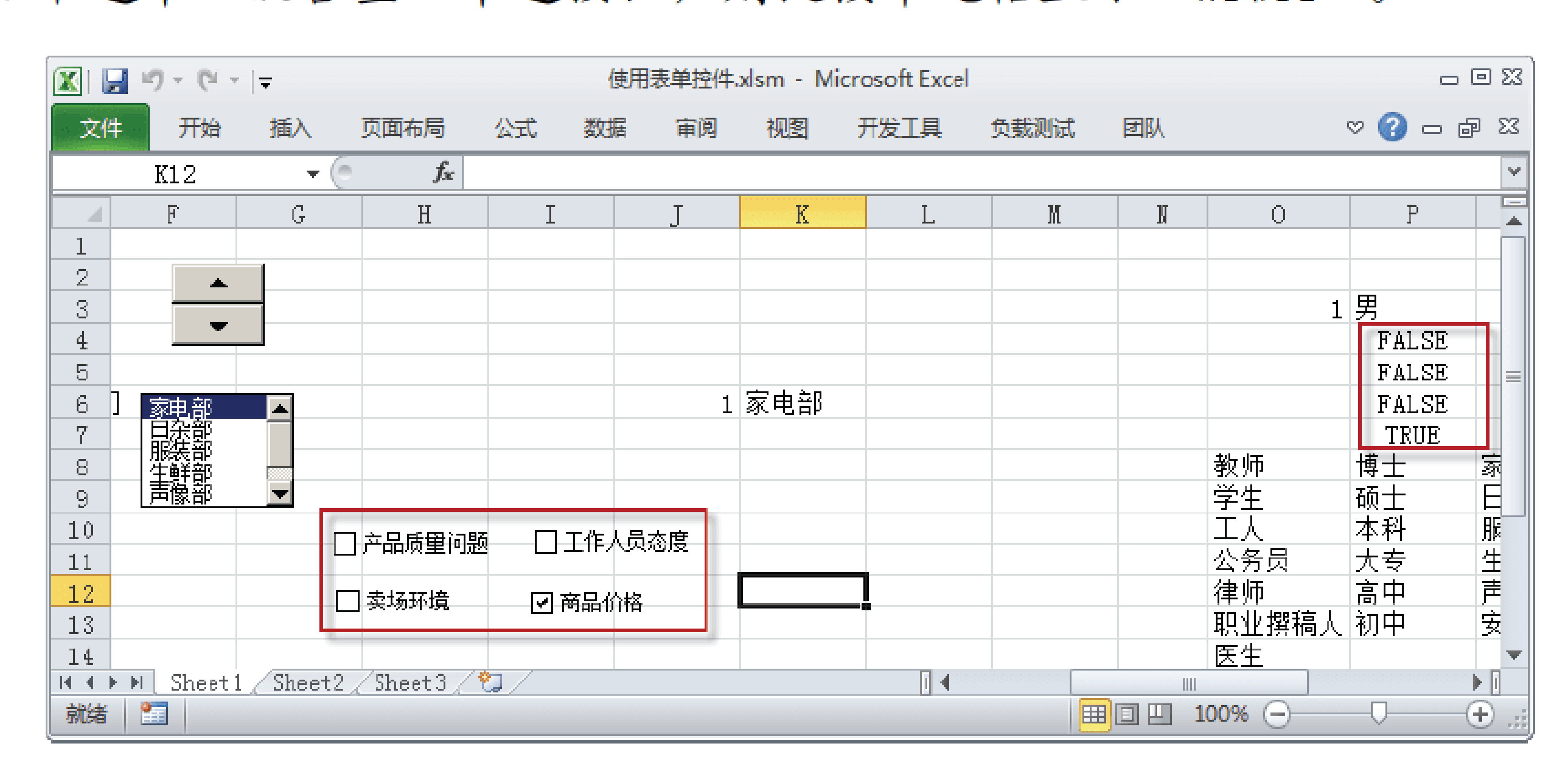


图 14.24　显示控件选择状态

14.2.6 “标签”控件

“标签”控件用于显示说明文字，如标题、题注或其他的提示信息。在工作表中使用该控件的优势在于，文字可以放置在工作表的任何位置而不受单元格的限制。在工作表中使用“标签”控件的方法如下。

（1）在工作表中添加“标签”控件，如图 14.25 所示。

（2）在控件中单击，修改控件中的标签文字。同时拖动控件边框上的控制柄调整控件的大小，拖动控件将其放置于需要的位置，如图 14.26 所示。

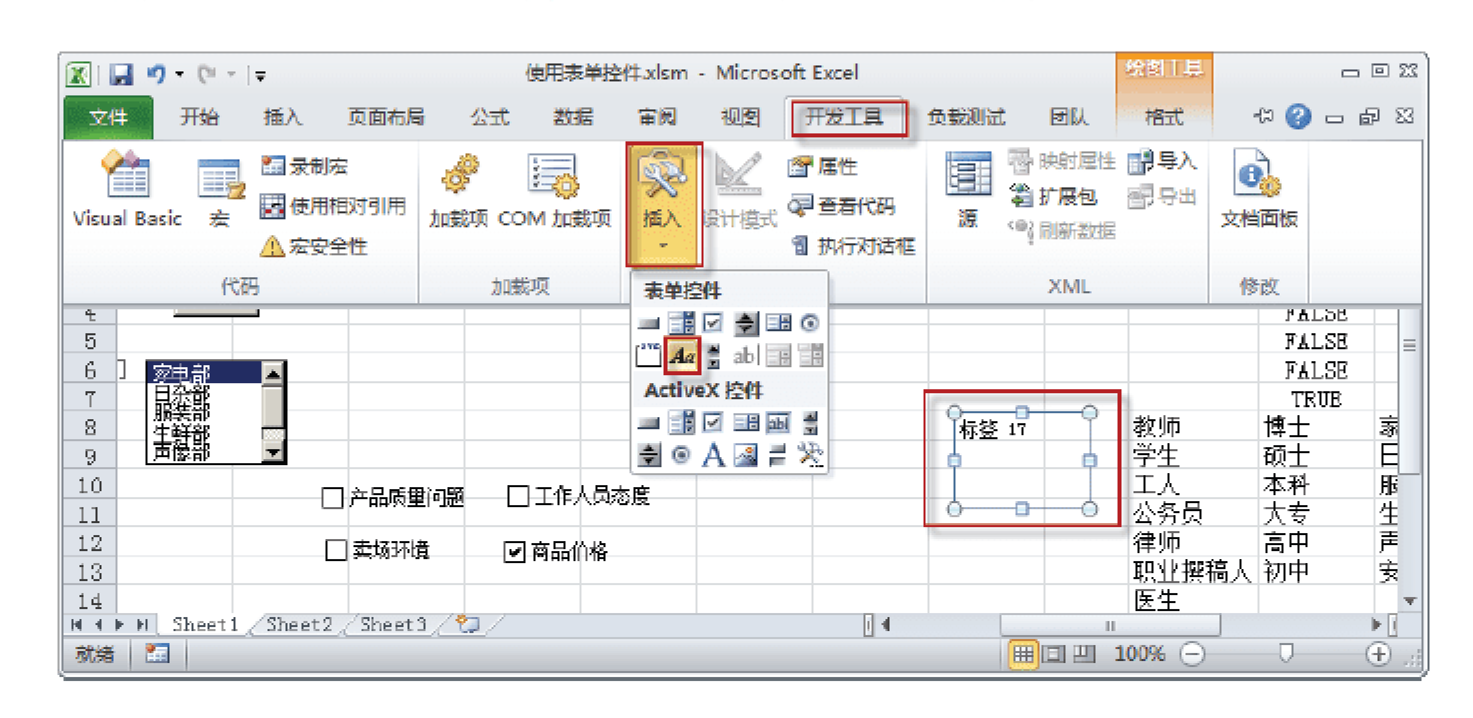


图 14.25　添加“标签”控件

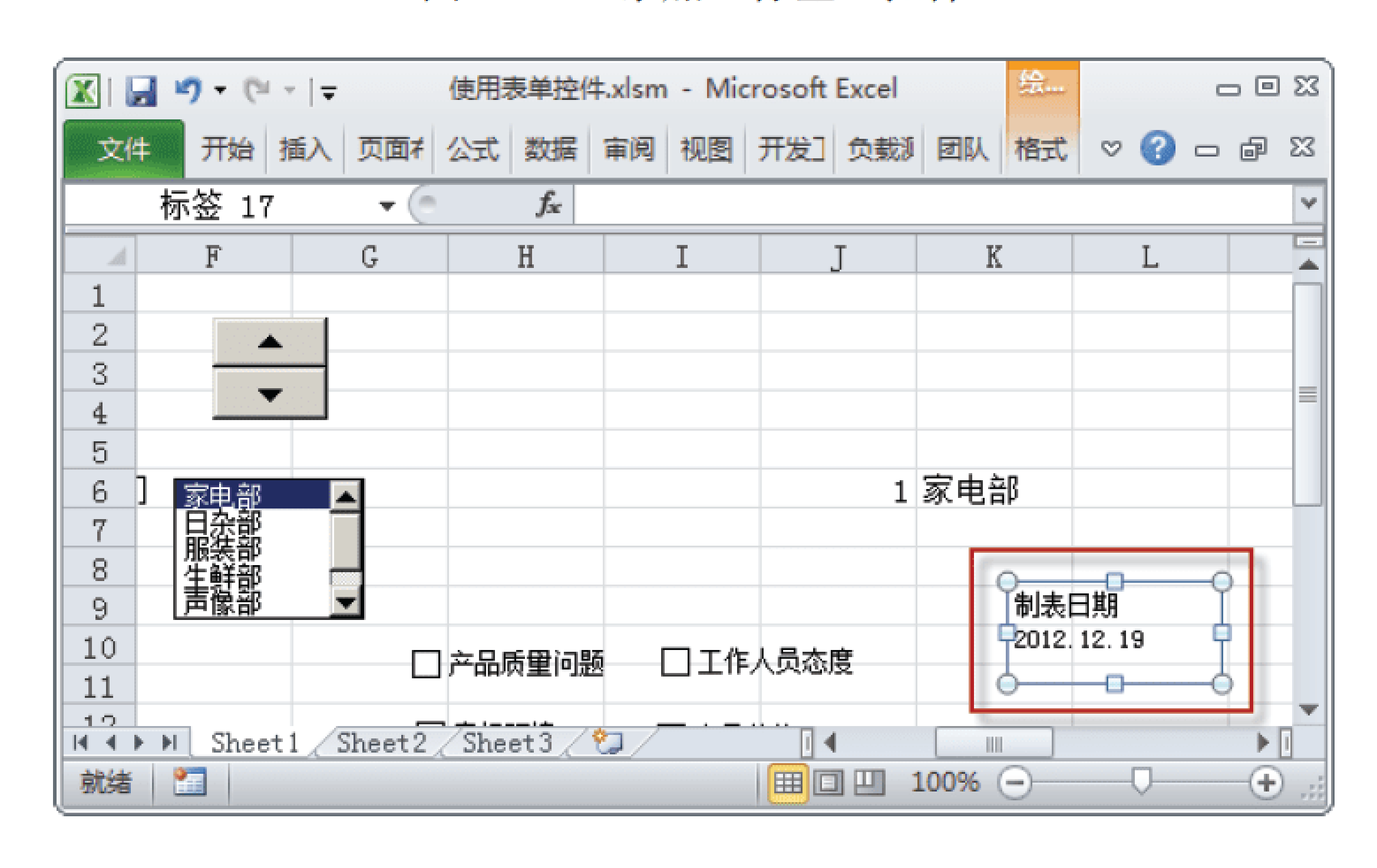


图 14.26　输入文字并调整大小和位置

提示： 控件显示的文字样式是可以改变的。选择控件上的文字后，使用功能区“开始”选项卡的“字体”和“对齐方式”组中的命令能够对控件文字的字体、字号、颜色和对齐方式等进行设置。

14.2.7 “按钮”控件

“按钮”控件是 Windows 应用程序中应用最多的一个控件，单击该控件能够启动程序。在 Excel 中，“按钮”控件常用于启动宏或 VBA 应用程序。在工作表中使用按钮控件的步

骤如下。

（1）创建一个宏，该过程将由“按钮”控件来启动。在 VBA 编辑器中创建一个模块，添加过程代码。这段过程代码用于显示用户的个人信息，其代码如下所示：

```
Sub 显示个人信息()
    MsgBox "您的姓名是：" & Sheet1.Range("C3") & Chr(13) _
    & "年龄：" & Sheet1.Range("E2") & Chr(13) _
    & "性别：" & Sheet1.Range("P3") & Chr(13) _
    & "婚否：" & Sheet1.Range("R3") & Chr(13) _
    & "职业：" & Sheet1.Range("T1") & Chr(13) _
    & "学历：" & Sheet1.Range("V3") & Chr(13) _
    , vbOKOnly, "您的基本个人信息"                    '显示基本个人信息
End Sub
```

（2）在工作表中添加一个“按钮”控件，如图 14.27 所示。此时，系统会自动打开“指定宏”对话框，在对话框的列表中选择上一步创建的宏，单击“确定”按钮关闭对话框，即可将宏指定给按钮，如图 14.28 所示。

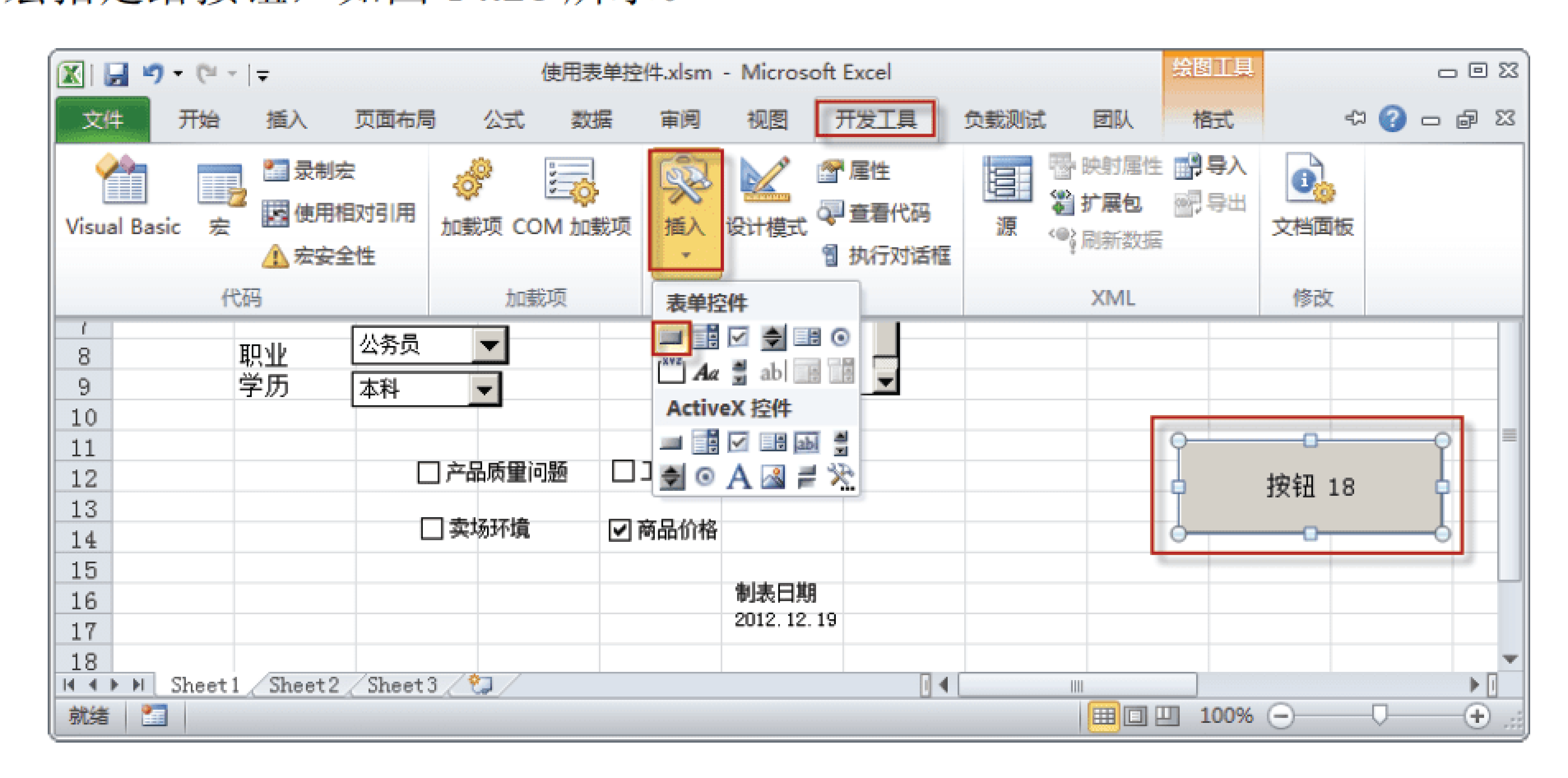


图 14.27　创建“按钮”控件

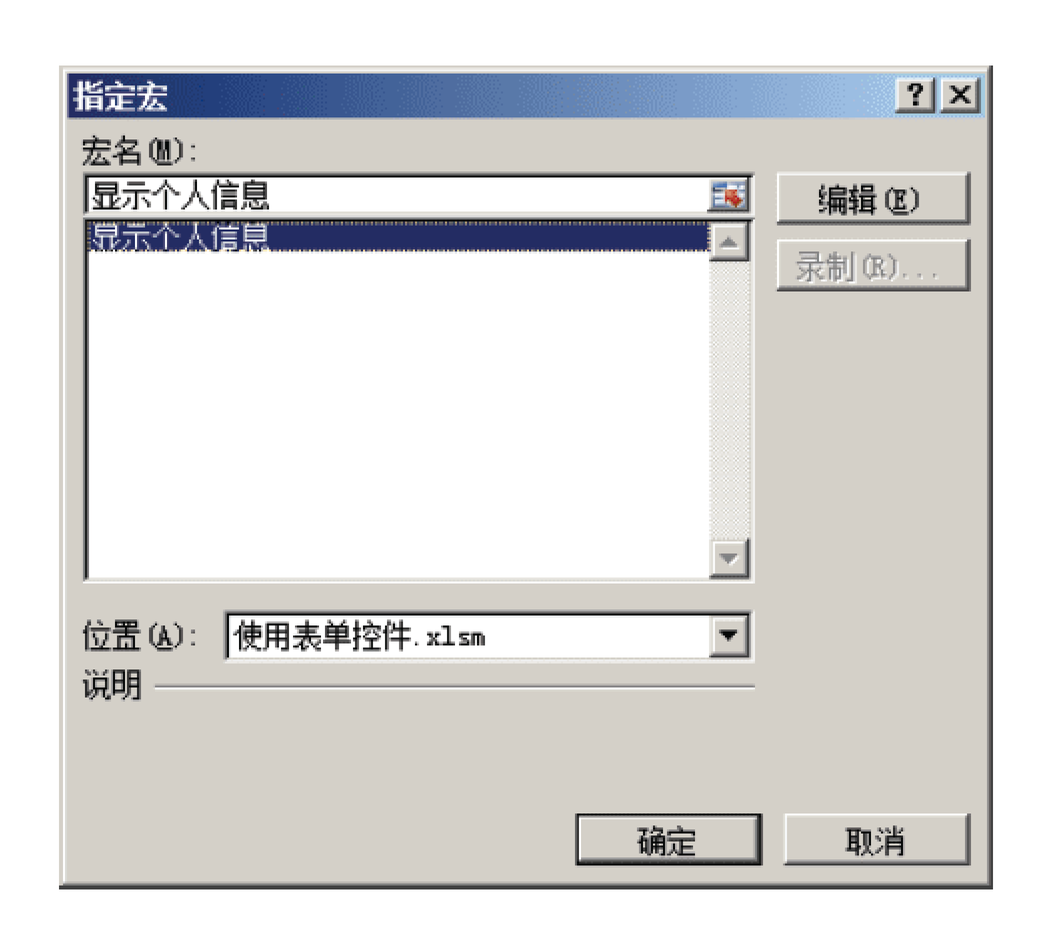


图 14.28　为按钮指定宏

（3）在控件被选择的情况下单击按钮，修改按钮标题文字后在工作表中单击取消按钮的选择。至此，本实例制作完成。在工作表中使用控件填写表格，单击该按钮将能够运行

程序，程序运行的效果如图 14.29 所示。

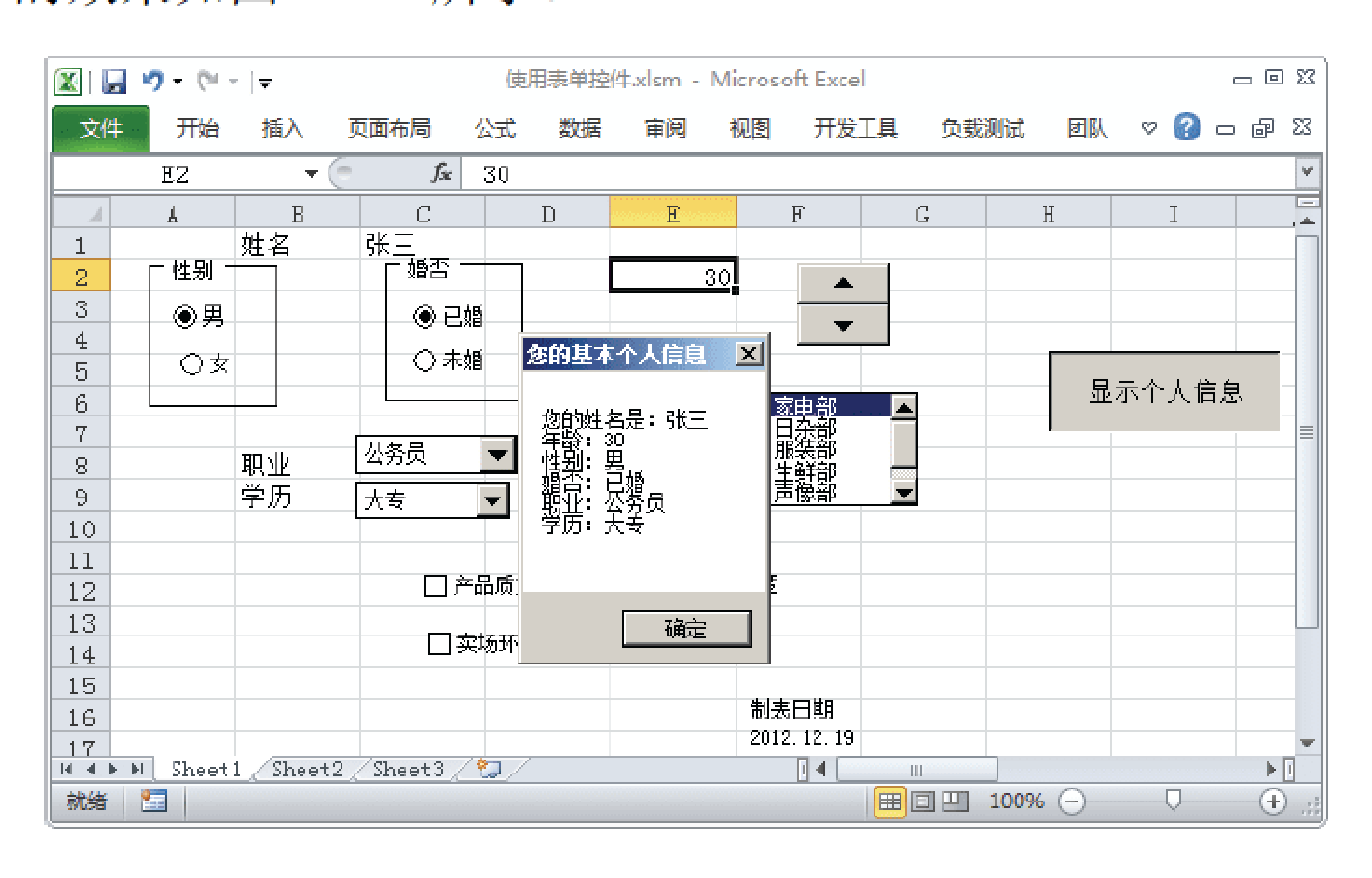


图 14.29　程序运行效果

提示：Excel 2010 的表单控件还包括“滚动条”控件，其使用与设置和“数值调节”控件相似，读者可以在第 3 章的练习中看到其使用实例，这里就不再赘述了。

14.3　使用 Excel ActiveX 控件

ActiveX 控件是由软件开发商开发的可重复使用的软件组件，可以被程序开发者作为预装配组件直接用于程序中。直接使用 ActiveX 控件，能够避免复杂或重复的编码过程，快速地向程序中添加需要的功能。

在 Excel 工作表中，可以像添加表单控件那样添加 ActiveX 控件，与表单控件相比，ActiveX 具有属性，能够在 VBA 编辑器的“属性”窗口中对控件属性进行设置。同时，ActiveX 控件具有事件过程，可以为事件过程编写事件代码，以实现事件响应。下面以一个基于 ActiveX 的数据录入系统的制作为例来介绍 ActiveX 控件的添加、属性设置和事件代码的编写过程。

14.3.1　添加控件和修改属性

与表单控件一样，Excel 2010 的 ActiveX 控件可以在“开发工具”选项卡的“插入”列表中进行选择。但 ActiveX 控件的属性设置需要通过更改控件的“属性”对话框中的属性值来实现。

（1）在 Excel 2010 中创建工作表，在工作表中绘制作为面板的图形。首先向工作表中添加一个“标签”控件，如图 14.30 所示。

（2）在控件上右击，选择“属性”命令，打开“属性”对话框设置控件的属性。这里修改控件的“BackStyle”属性，将控件背景设置为透明，更改 Caption 属性的设置标签文字，如图 14.31 所示。

注意：ActiveX 控件的属性设置与表单控件的属性设置不同，需要在“属性”面板中进行。一般情况下，属性值也可以使用程序来改变。如果读者不了解一个控件有哪些属性，可以参考“属性”面板中的设置项。

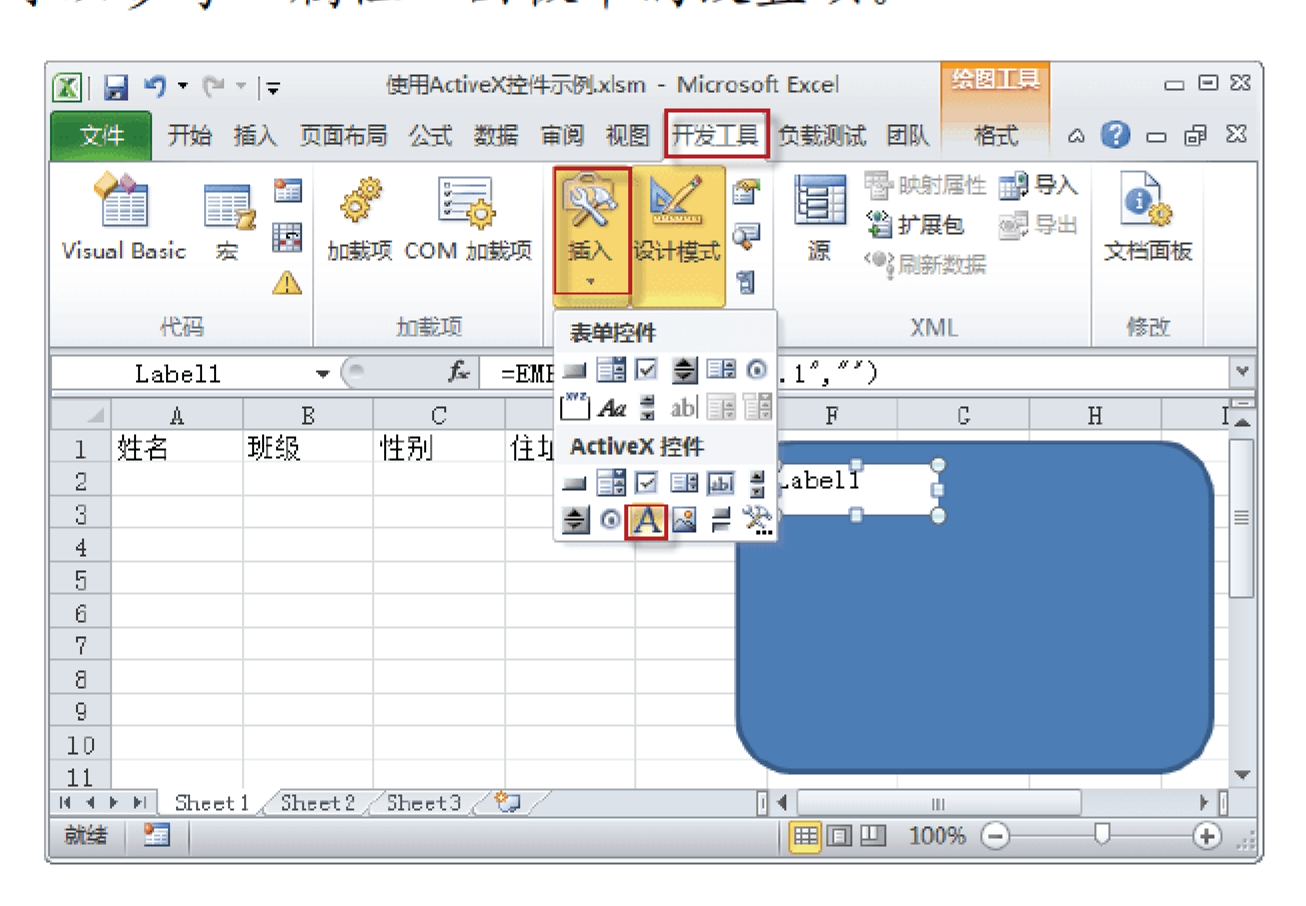


图 14.30　添加“标签”控件

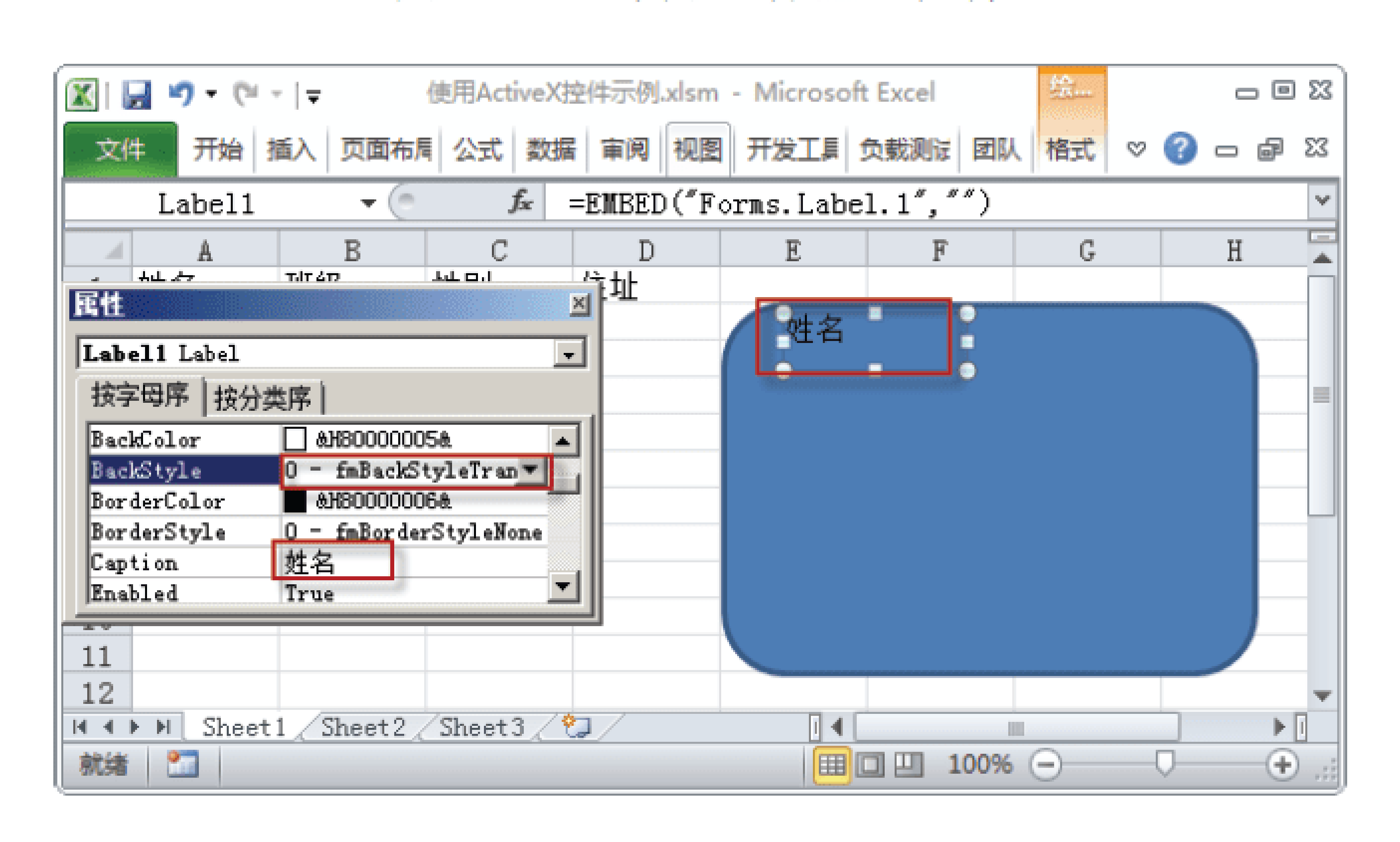


图 14.31　设置控件属性

（3）在工作表中添加一个“文本框”控件，如图 14.32 所示。该控件的属性使用默认值即可。

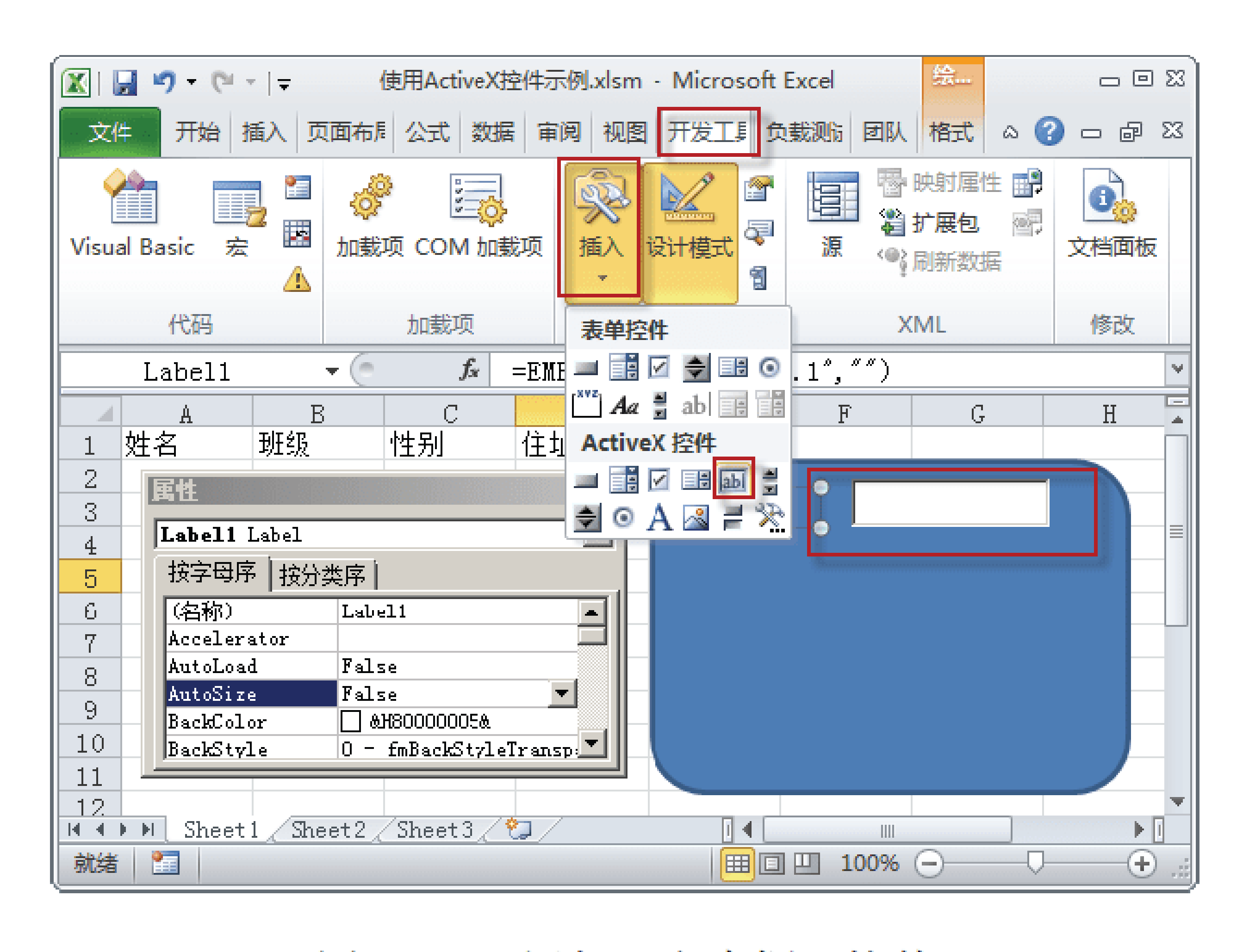


图 14.32　添加“文本框”控件

（4）在工作表中再添加一个“标签”控件和一个“组合框”控件，如图 14.33 所示。

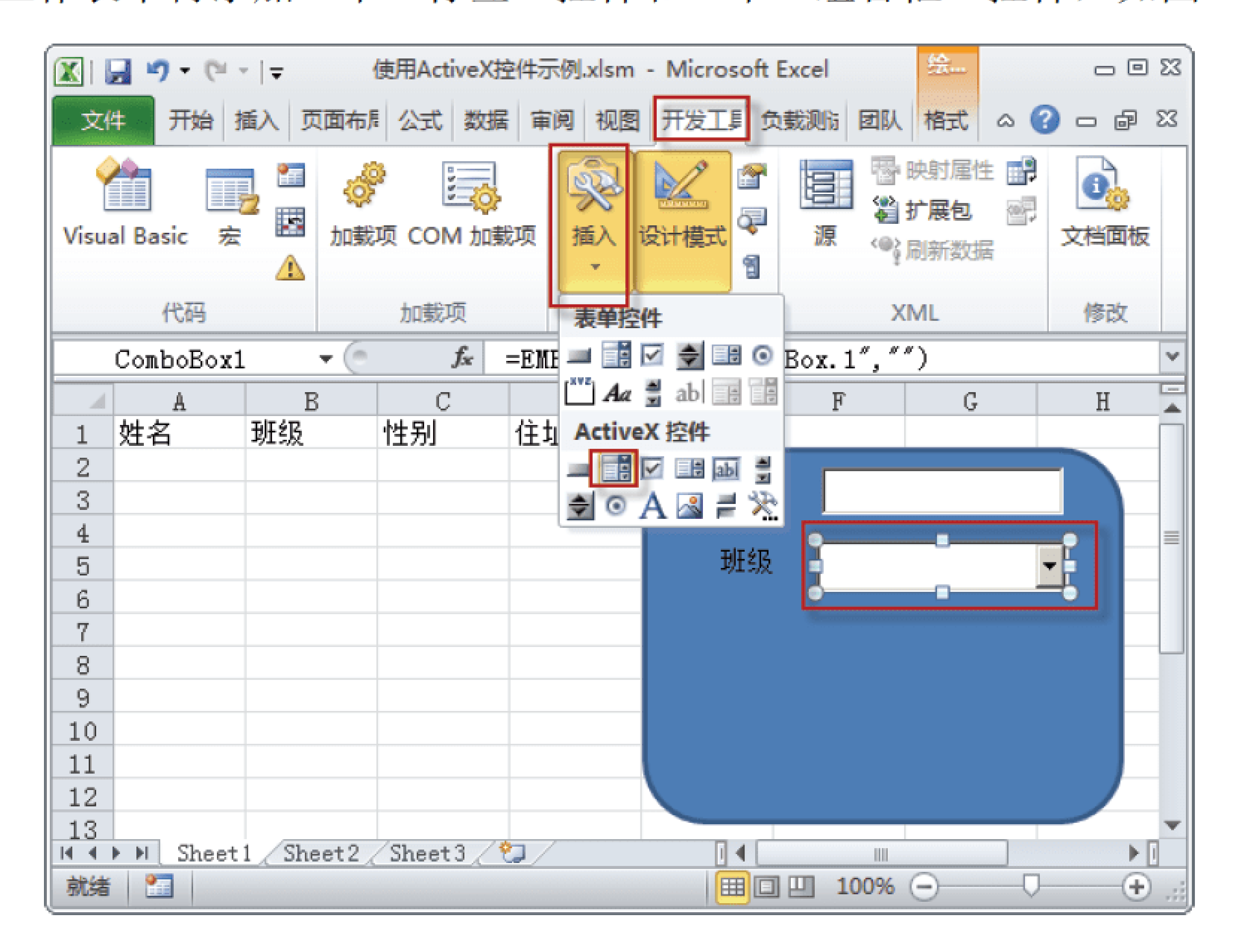


图 14.33　添加控件

（5）将“标签”按钮复制一个，并将其标题修改为“性别”。同时再添加 2 个“选项按钮”，如图 14.34 所示。

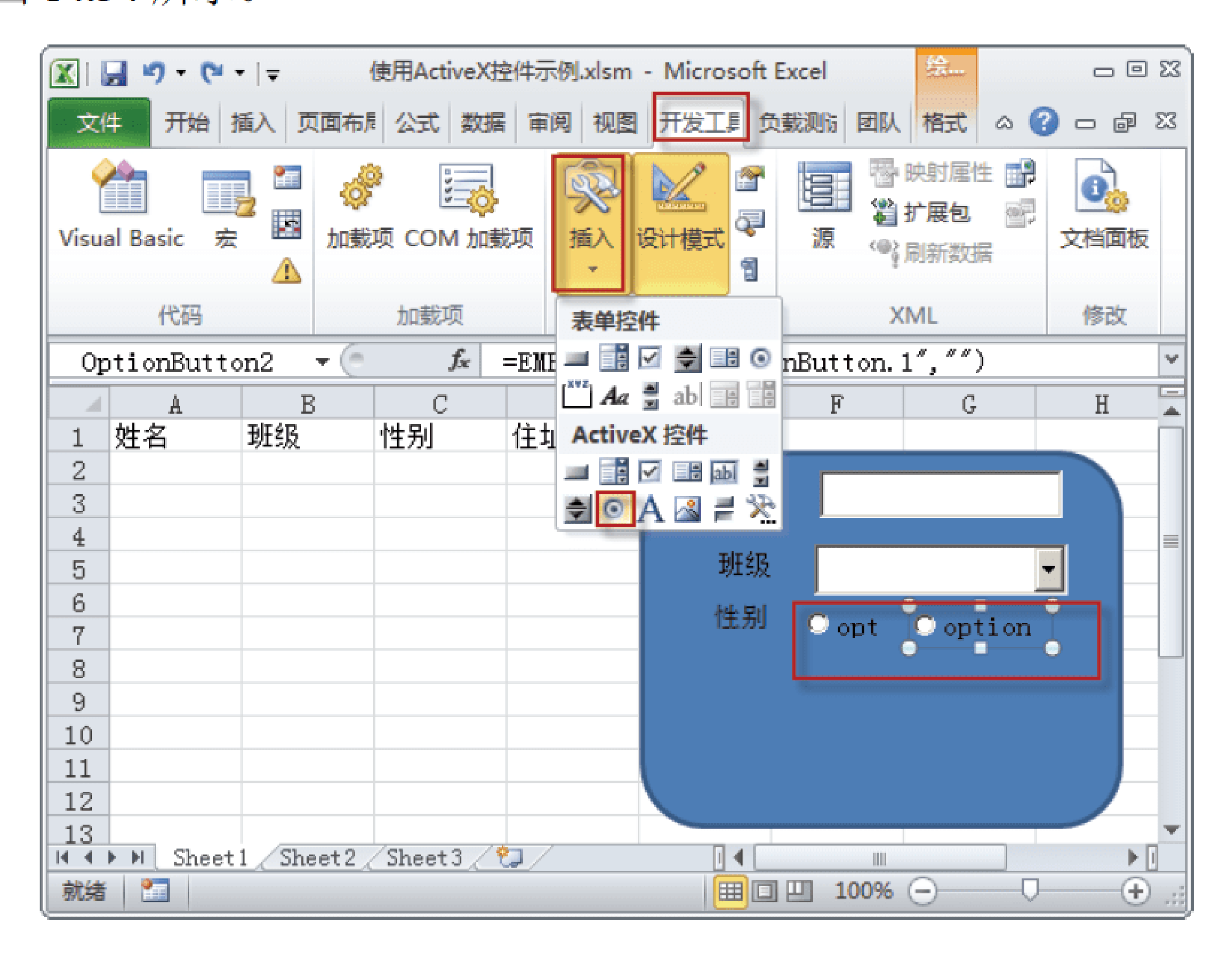


图 14.34　添加两个“选项按钮”

（6）设置“选项”按钮的 Caption 属性和 BackStyle 属性，如图 14.35 所示。

（7）采用相同的方法，向工作表中添加一个“标签”控件、一个“文本框”控件和两个“命令按钮”控件。在“属性”对话框中分别修改“标签”控件和两个“命令按钮”控件的 Caption 属性。完成控件添加后的效果如图 14.36 所示。

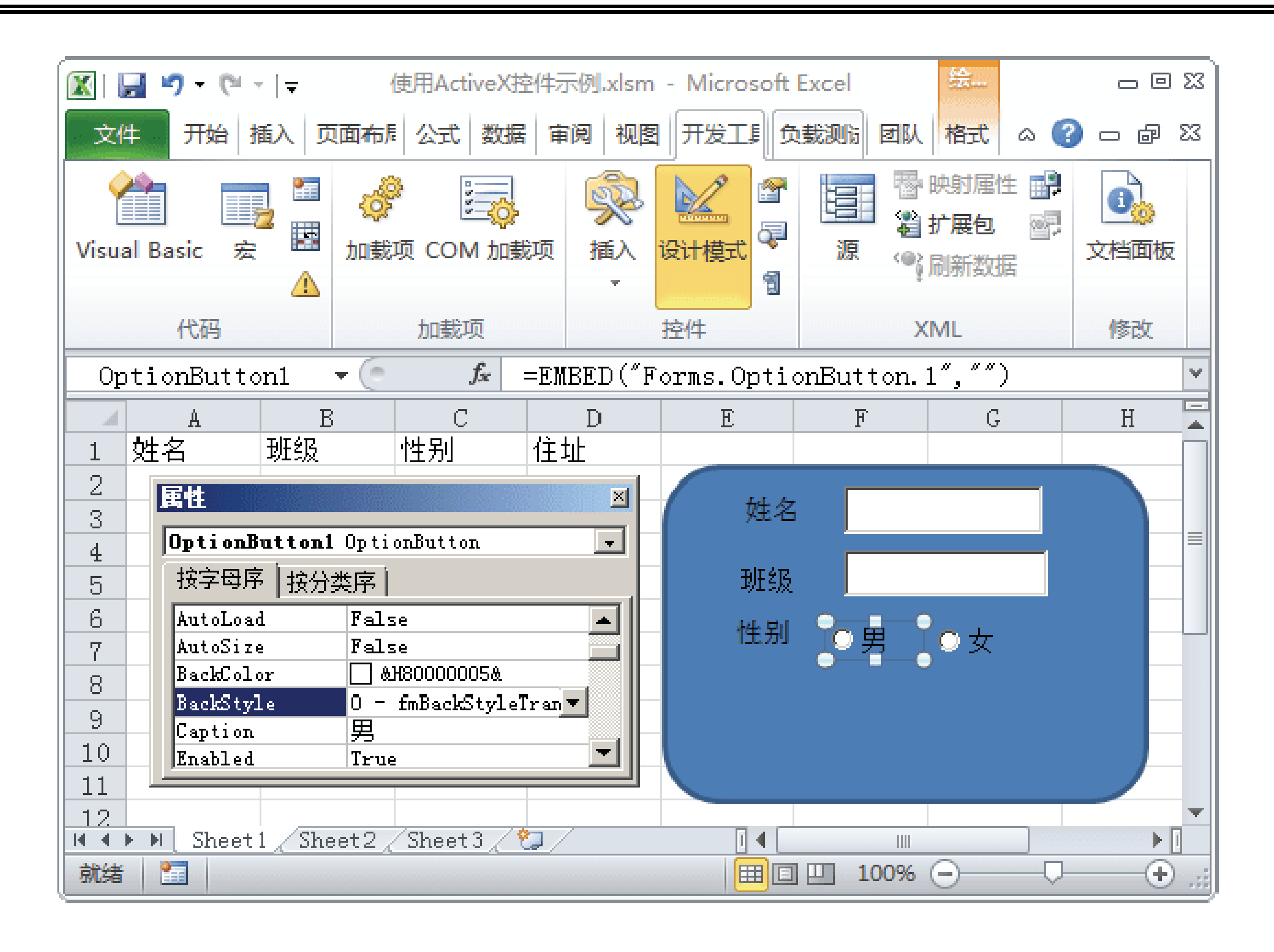


图 14.35　设置控件属性

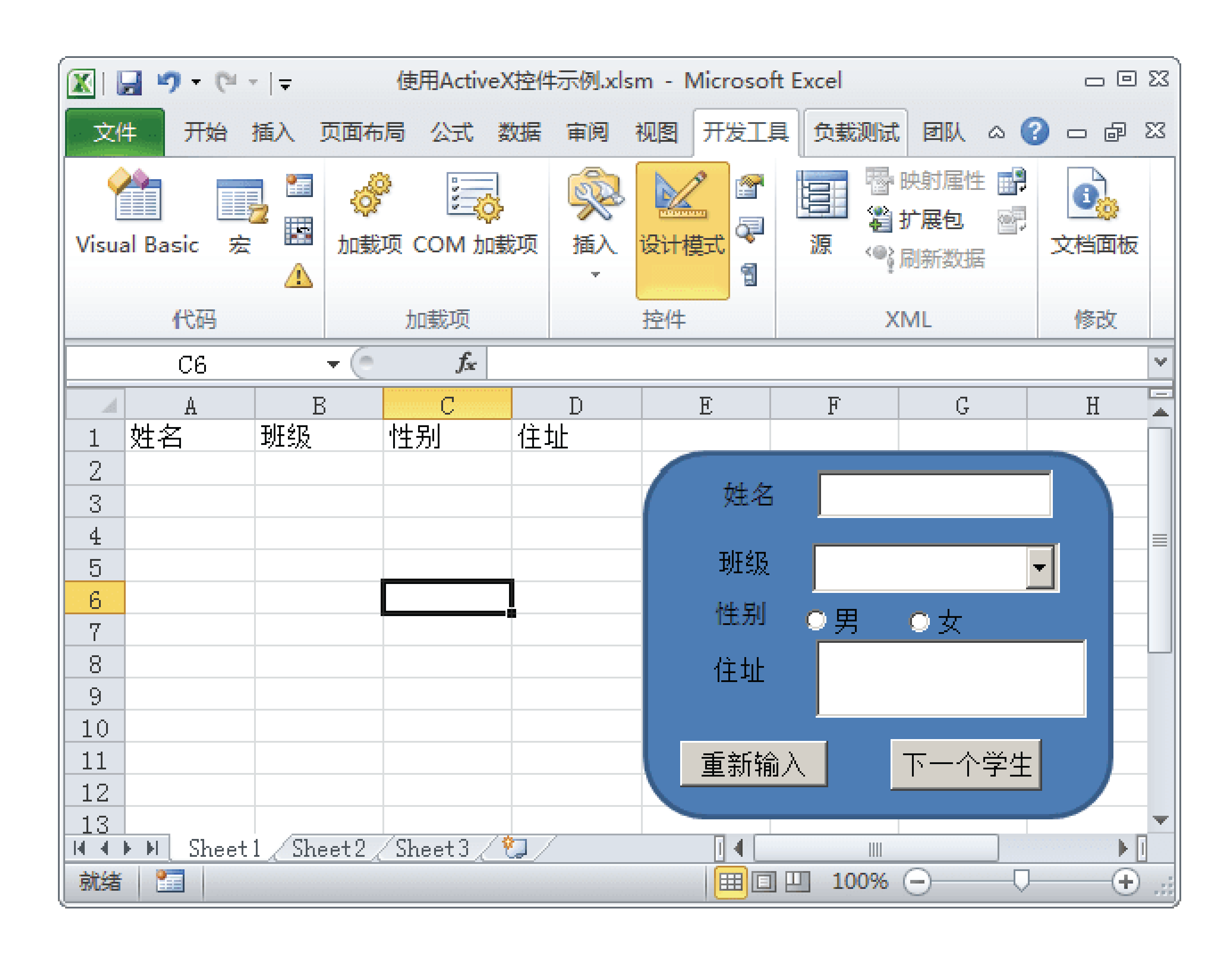


图 14.36　完成控件添加后的效果

14.3.2　为控件添加程序代码

与表单控件使用不同的是，ActiveX 控件的功能是通过事件响应程序来实现的。在工作表中添加 ActiveX 控件后，必须为其添加事件代码，也就是告诉控件在事件发生时该做什么。下面介绍对控件编程的过程。

（1）打开 Visual Basic 编辑器，在工程管理器中双击“sheet1”链接，打开“代码”窗口，在窗口中首先添加如下代码：

```
Public n  As Integer                          '声明一个全局变量
Private Sub Worksheet Activate()
    n = 2                                     '初始化变量 n
    Range("a2").Select                        '选择单元格
    If ComboBox1.ListCount = 0 Then           '判断列表框中是否有数据
        With ComboBox1                        '为“组合框”控件添加列表项
            .AddItem "七年级（1）班"
```

```
08              .AddItem "七年级（2）班"
09              .AddItem "七年级（3）班"
10              .AddItem "七年级（4）班"
11              .AddItem "七年级（5）班"
12          End With
13      End If
14  End Sub
```

提示：代码的第一行声明了一个全局变量 n，该变量表示工作表当前的行号，该变量在其他事件代码中也会被使用。这里的 Worksheet_Activate()事件是在工作表变为活动工作表时触发，因此其事件响应程序常用于对对象的初始化。填写数据是从工作表的第 2 行开始，因此变量 n 设置为 2。第 05 ~ 13 行使用 With 结构来初始化“组合框”控件。该控件必须使用 Add 方法来添加列表项，这个过程必须在工作表激活时进行。

（2）在“代码”窗口中接着输入如下代码。为“姓名”文本框添加 Change 事件程序。该文本框用于姓名的输入，如图 14.37 所示。

```
01  Private Sub TextBox1_Change()
02      Range("a" & n).Value = TextBox1.Text    '将文本框的内容填入单元格
03  End Sub
```

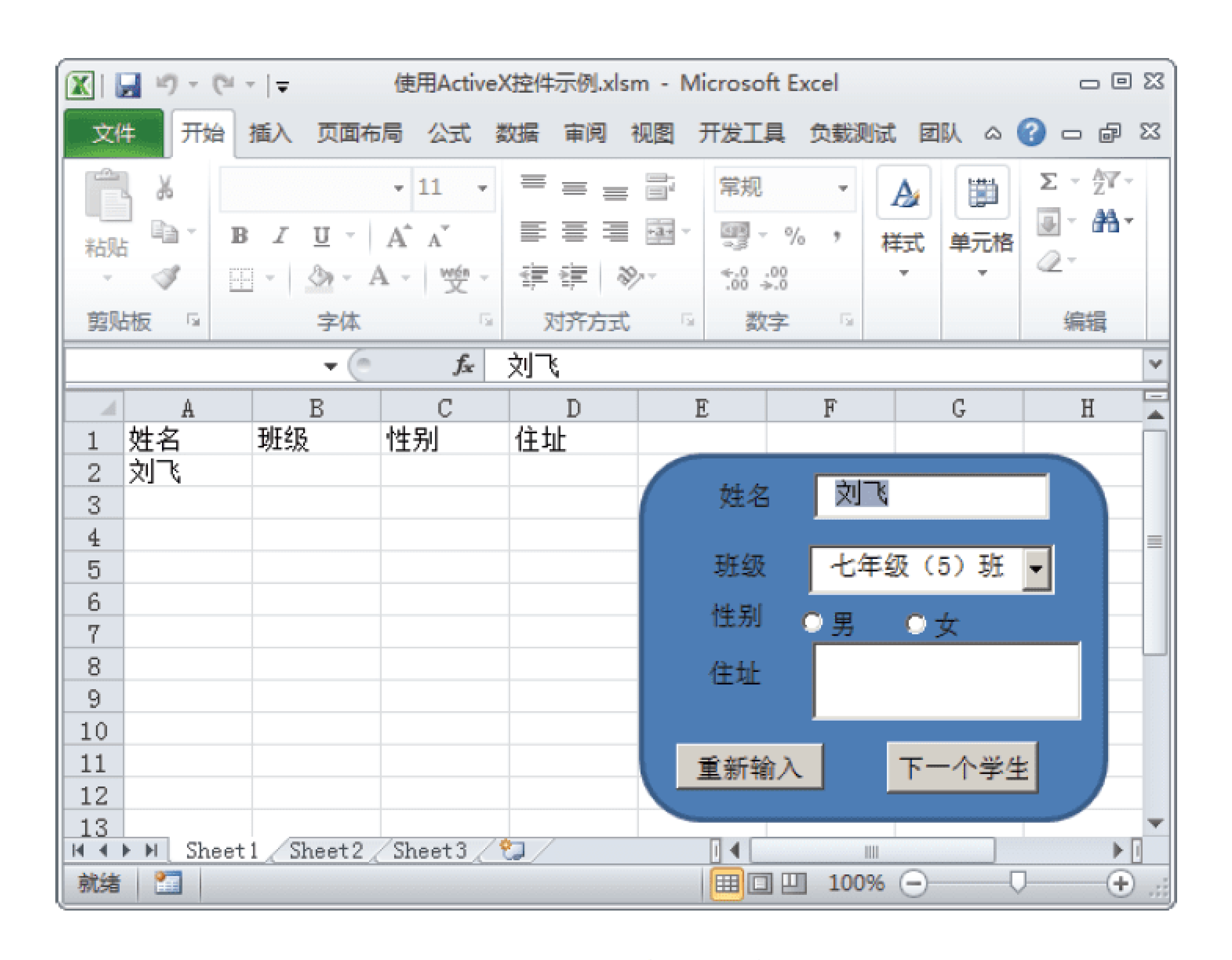


图 14.37　输入姓名

提示：Textbox1_Change()事件是在文本框中内容发生改变时产生。“文本框”控件的 Text 属性保存着文本框中的内容，第 02 行将其写入指定的单元格中。这里的变量 n 值决定了单元格所在的行。

（3）在“代码”窗口中输入如下代码。为“班级”“组合框”控件添加 Change 事件代码。当控件中选项发生改变时，选择的内容将填入指定的单元格中，如图 14.38 所示。

```
01  Private Sub ComboBox1_Change()
```

```
        Range("b" & n).Value = ComboBox1._
        List(ComboBox1.ListIndex)          '将“组合框”控件中的选择填入单元格中
    End Sub
```

🔔提示：ComboBox1_Change()事件在组合框选项发生改变时被触发。在程序中，ComboBox1.ListIndex 语句获得“组合框”控件列表项的索引号，List 方法根据索引号得到当前的选项。

（4）在“代码”窗口中输入如下代码。

```
    Private Sub OptionButton1 Click()
        If OptionButton1.Value = True Then        '判断是否选中了第一个单选按钮
            Range("c" & n).Value = "男"            '是，则单元格写入“男”
        Else
            Range("c" & n).Value = "女"            '没有选择，单元格写入“女”
        End If
    End Sub
    Private Sub OptionButton2 Click()
        If OptionButton2.Value = True Then        '判断是否选中了第二个单选按钮
            Range("c" & n).Value = "女"            '是，则单元格写入“女”
        Else
            Range("c" & n).Value = "男"            '没有选择，单元格写入“男”
        End If
    End Sub
```

完成代码输入后，当这 2 个单选按钮被选中时，将选择的性别填入单元格中，如图 14.39 所示。

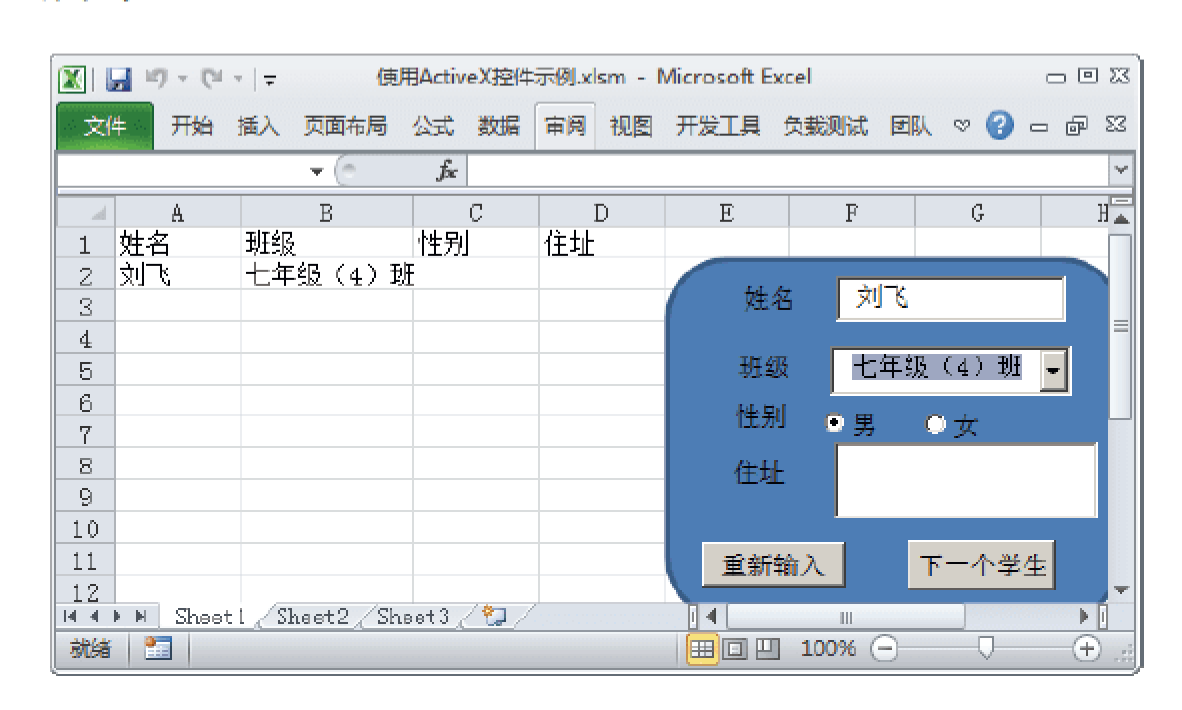


图 14.38　选择班级

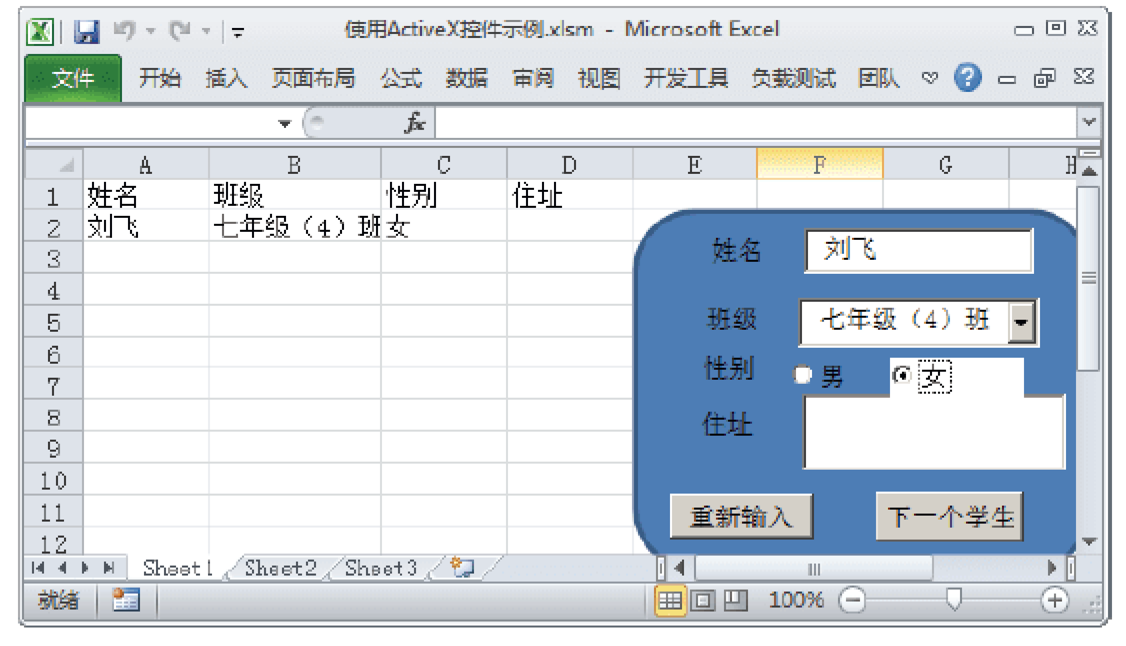


图 14.39　选择性别

🔔提示：OptionButton2_Click()事件是在“单选按钮”控件被单击时触发。在事件程序中，使用 If...Else 结构来判断单击的是哪个按钮。如果 Value 属性为 True，说明按钮被选中，则向单元格填写选择的内容。

（5）在“代码”窗口中输入如下代码。在“地址”的文本框中输入的内容可以填入指定的单元格，如图 14.40 所示。

```
    Private Sub TextBox2_Change()
        Range("d" & n).Value = TextBox2.Text    '将文本框内容填入单元格
    End Sub
```

（6）在“代码”窗口中输入如下代码。当单击“重新输入”按钮时，将单元格和文本

框清空，如图 14.41 所示。

```
Private Sub CommandButton1_Click()        '单击按钮，所有单元格和文本框清空
    Range("a" & n).Value = ""              '“姓名”单元格清空
    Range("b" & n).Value = ""              '“班级”单元格清空
    Range("c" & n).Value = ""              '“性别”单元格清空
    Range("d" & n).Value = ""              '“地址”单元格清空
    TextBox1.Text = ""                     '“姓名”文本框清空
    TextBox2.Text = ""                     '“住址”文本框清空
End Sub
```

这里，CommandButton1_Click()是在单击按钮时触发的事件。

（7）在“代码”窗口中输入如下代码。

```
Private Sub CommandButton2_Click()
    n = n + 1                                         '变量加 1 指向下一行
    Range("a" & n).Select                             '选择单元格
End Sub
```

当单击“下一个学生”按钮后，选择下一行第一个单元格，如图 14.42 所示。此时可继续进行第二个学生信息的输入。

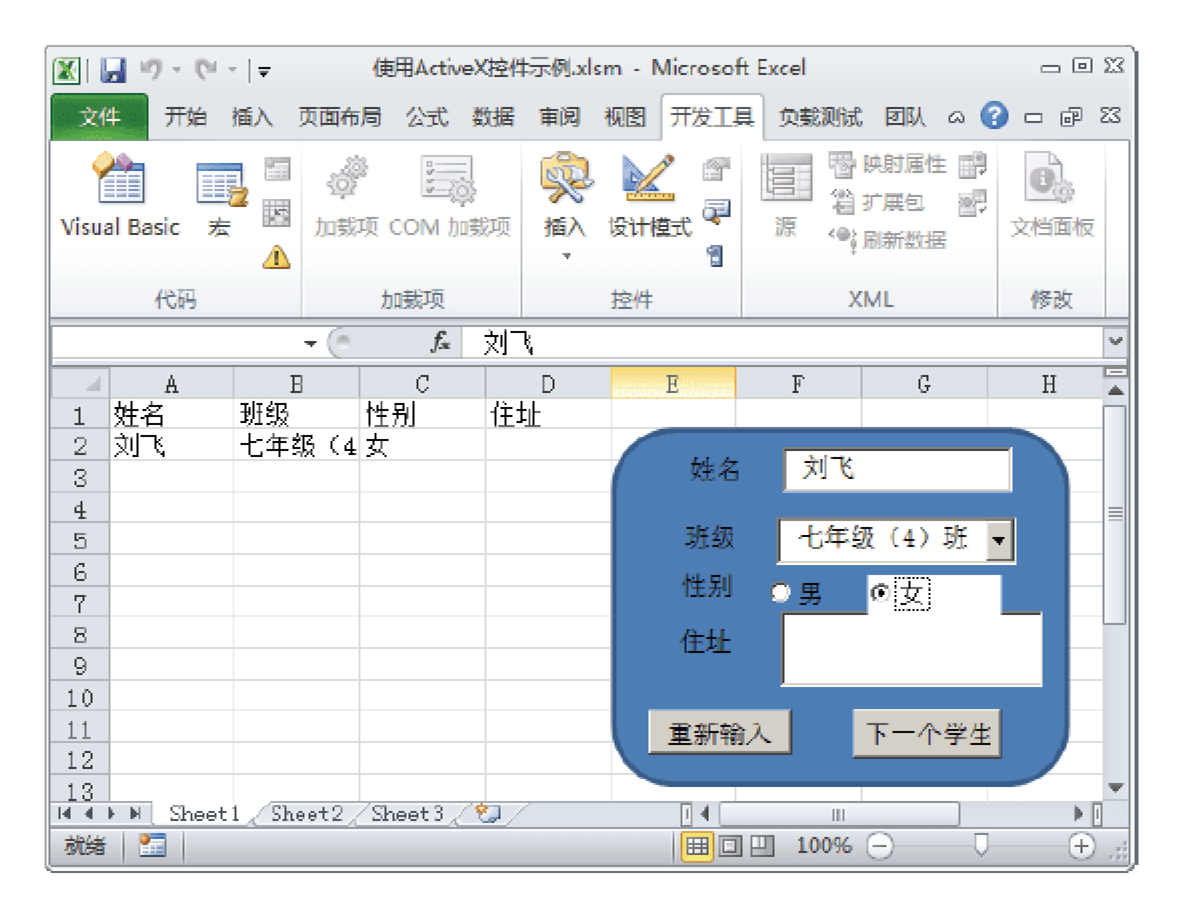


图 14.40 输入地址

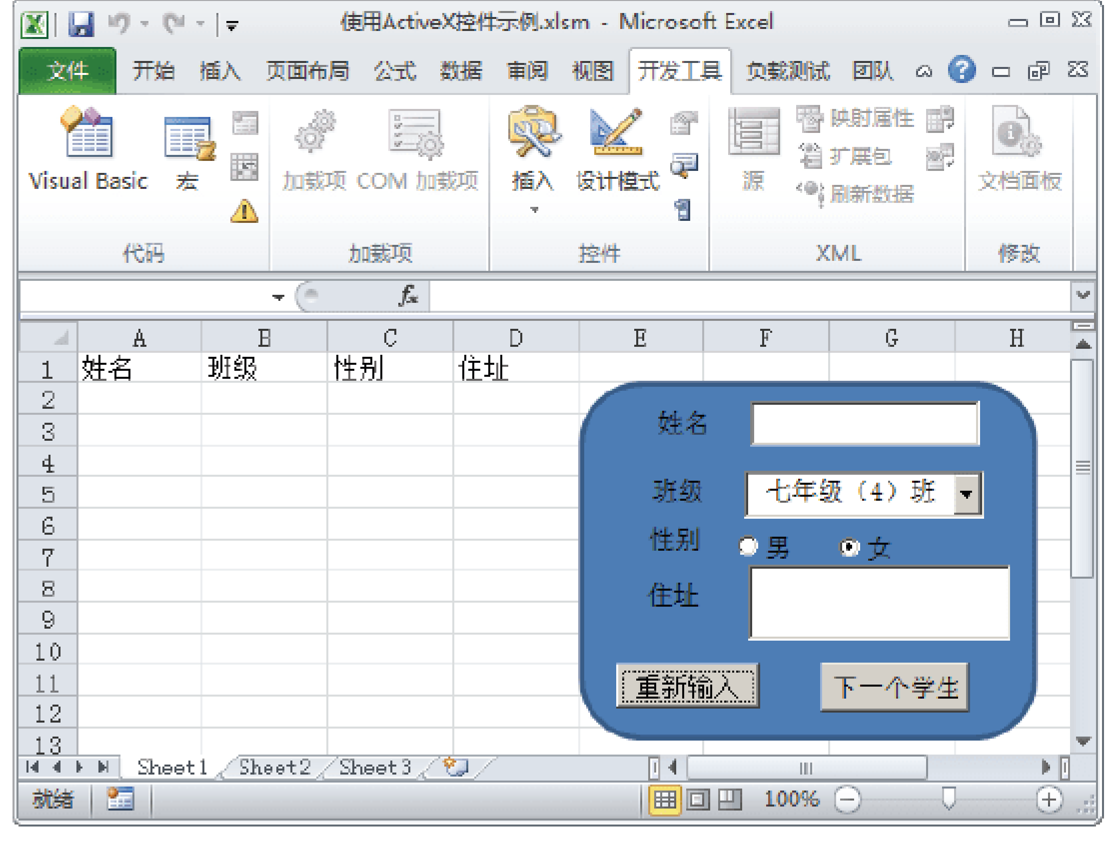


图 14.41 清空单元格和文本框

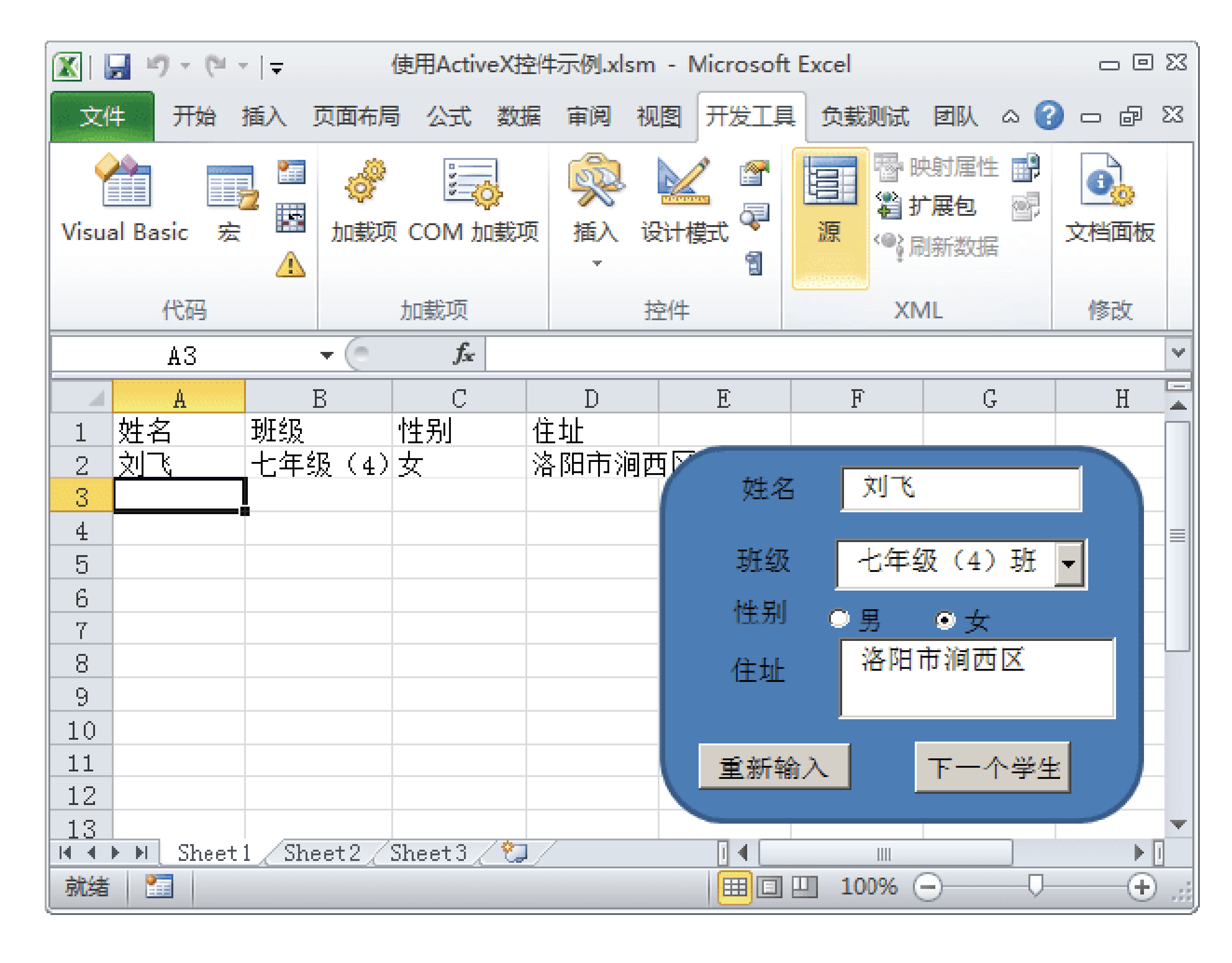


图 14.42 选择下一行第一个单元格

14.4　小　　结

在 Excel 2010 工作表中使用控件，能够极大地方便应用程序的控件，更为方便地实现人机对话，使应用程序更专业。本章以实例的形式介绍了 Excel 2010 的表单控件和 ActiveX 控件在工作表中的使用。通过本章的学习，读者将掌握这两类控件的一般使用方法，理解它们在用法上的区别，能够在应用程序中熟练使用它们。

操作界面友好是设计操作界面的一个重要原则，界面友好意味着对于任何一个新用户在接触软件时能够很快掌握界面操作。在设计界面时，还需要考虑界面控件的布局以及界面和控件的色彩搭配等诸多因素。一个良好的操作界面应该做到控件布局排列有序、界面色调明快大方且功能设置方便合理，易于使用户接受。

14.5　本 章 习 题

1．在工作表中输入文字，而文字又不希望放置于单元格中，可以选择下面哪种控件？（　　）

A．“单选按钮”控件　　　　B．“标签”控件

C．“列表框”控件　　　　D．“复选框”控件

2．下面哪个操作能够修改“按钮”表单控件的标题？（　　）。

A．直接单击选择该控件后进行修改　　　　B．双击控件后直接修改

C．右击选中该控件后再进行修改　　　　D．拖动光标框选控件后进行修改

3．要改变“标签”ActiveX 控件应该使用下面哪种方法？

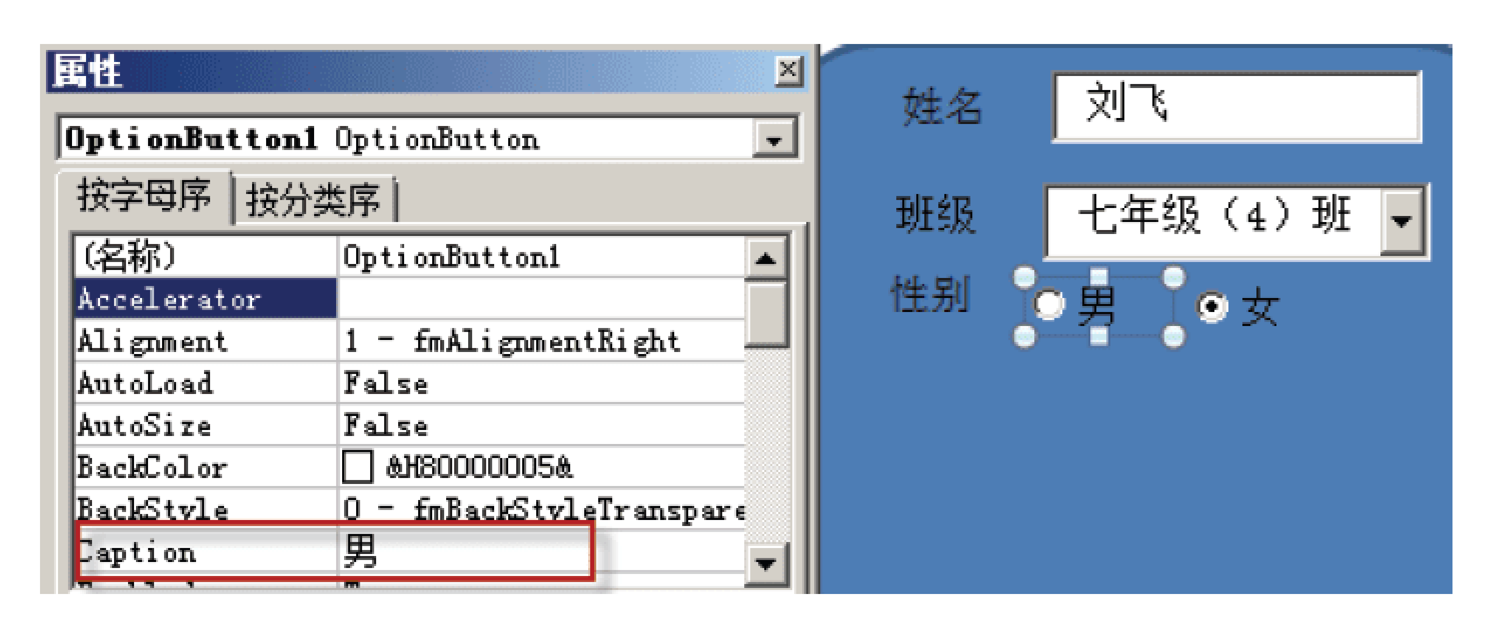


图 14.43　改变“标签”

A．在控件的文字上单击后修改

B．在“属性”对话框中修改 Caption 属性

C．在“属性”对话框中更改 Value 属性

D．双击控件即可直接修改文字

4．使用表单控件设置和更改选择单元格的文字样式。要求使用“组合框”控件选择文字字体，“滚动条”控件设置文字大小，“单选按钮”控件设置文字是否加粗，“按钮”控件控制文字样式的更改。

【提示】在工作表中添加控件，并对控件进行设置。由于表单控件本身是无法像 ActiveX 控件那样添加事件代码的，因此需要编写一个过程来获取控件在单元格中的值，然后将该过程代码指定到“按钮”控件上，使用“按钮”控件来启动程序实现对文字样式的改动。

5．在工作表中使用“组合框”ActiveX 控件直接对选择单元格中文字应用指定样式，使用“按钮”控件将文字样式恢复为默认样式。

【提示】使用控件进行控制比较简单，在工作表中添加控件，并设置好控件的外观属性，然后为 Sheet1 工作表添加事件代码。这里首先为“组合框”控件添加选项，该功能应该在工作表的 Sheet1 的 Activate 事件过程中实现。然后，直接为“组合框”控件添加 Change 事件代码，为“按钮”控件添加 Click 事件代码即可。

第 15 章　自定义 Excel 用户窗体

窗体是用户界面的基本表现形式，可以根据需要自定义用户窗体，在窗体中添加各种控件用以实现交互界面的便捷性与友好性，在自定义窗体上添加标准控件以及在窗体上使用附加的 ActiveX 控件，其中常用的控件有标签控件、文本框控件、组合控件、列表框控件以及图像控件等。本章要学习的内容和目标如下所示：

- 掌握 Excel 2010 用户窗体的添加方法，以及设置窗体的属性；
- 掌握 Excel 2010 中标准控件的使用方法；
- 掌握 Excel 2010 附加控件 ListView 控件、ImageList 控件和 TreeView 控件的使用方法。

15.1　使用 Excel 窗体

为了方便与用户间的交互，Excel 2010 自带有内置窗体，如 MsgBox 和 InputBox 函数产生的对话框就是简单的用户窗体。在实际操作中，针对不同的应用需要，用户往往需要设计自己的用户界面，这就必然要使用 VBA 的用户窗体，以窗体搭载控件来实现需要的操作。本节将首先介绍 Excel VBA 中用户窗体的添加、属性的设置和对窗体编程的方法。

15.1.1　添加用户窗体

用户界面实际上就是由窗体和窗体中的控件所构成的操作环境，窗体在界面中成为了控件的载体。因此，在设计用户界面时首先需要创建用户窗体。在 Excel 中，创建用户窗体可以在 VBA 编辑器中操作完成。下面以创建一个用户窗体为例来介绍具体的操作步骤。

（1）创建一个新的 Excel 工作簿，按“Alt+F11”组键打开 Visual Basic 编辑器。选择“插入”→“用户窗体”命令即可向工程中添加一个用户窗体，如图 15.1 所示。

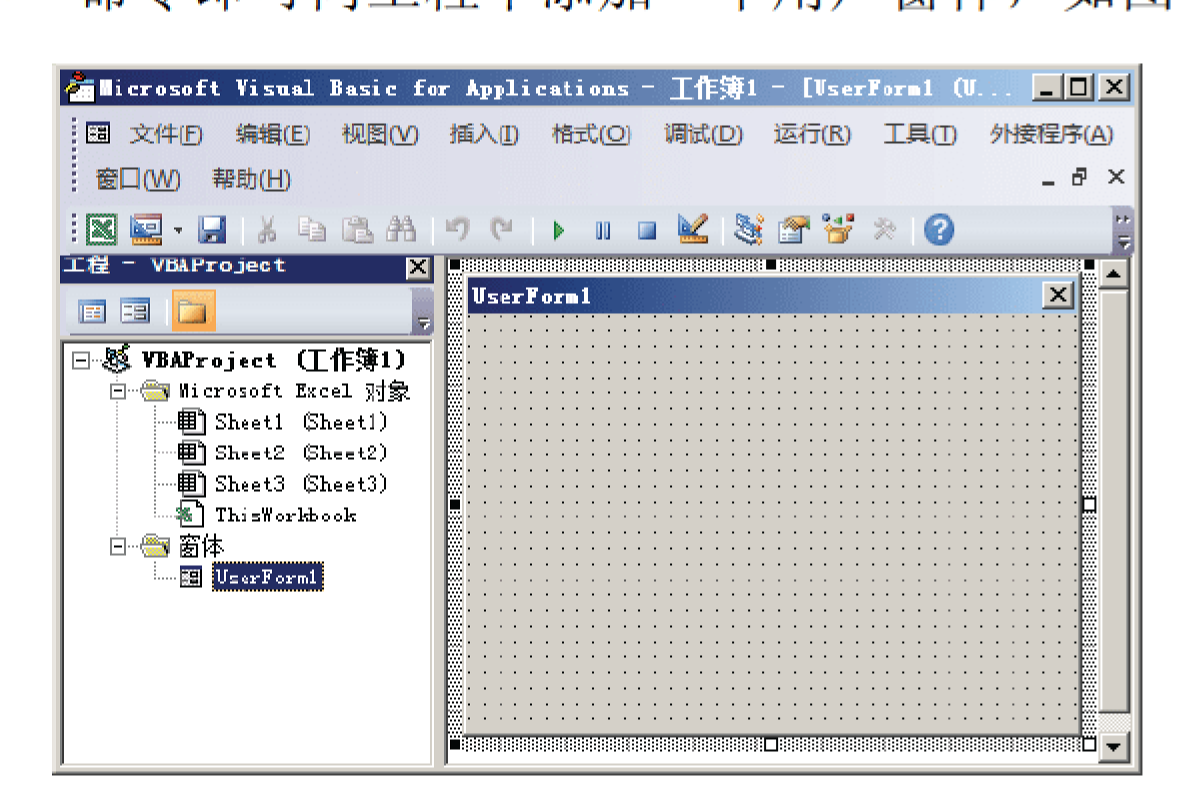


图 15.1　插入一个用户窗体

（2）右击工程资源管理器，在弹出的快捷菜单中选择“插入”→“用户窗体”命令，如图 15.2 所示，同样能够创建一个用户窗体。

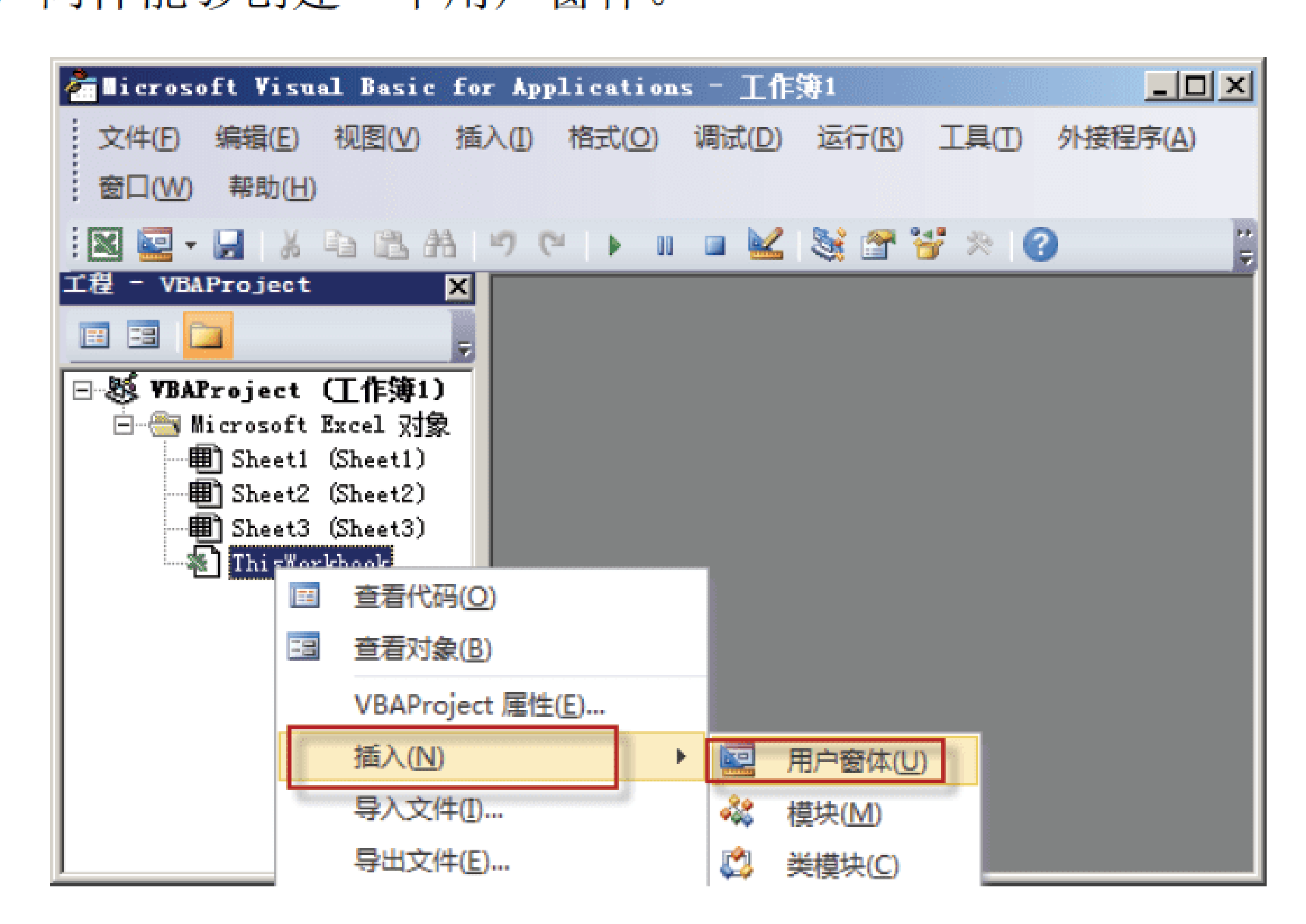


图 15.2　使用工程资源管理器来创建用户窗体

注意：用户窗体一旦被创建，就存在于 Excel 工作簿中，即使窗体没有加入程序代码，也同样能够被启动。启动用户窗体的方法是：选择“运行”→“启动用户子程序/用户窗体”命令或按 F5 键。

15.1.2　设置窗体的属性和事件

用户窗体是 VBA 的一个对象，窗体的属性决定了窗体的外观和行为。插入窗体时，VBA 会使用默认的属性值创建一个空白窗体。在程序设计时，可以重新设置窗体的属性值来改变其默认特征。窗体的属性可以在编写程序前在“属性”对话框中设置，也可以通过编写程序来改变。

在程序运行时，有时需要更改窗体的标题栏文字，如将标题栏文字设置为打开的工作表名，就像 Excel 2010 程序窗口那样。此时，可以通过修改窗体的 Caption 属性值来实现。窗体的 Caption 属性值决定了窗体标题栏上显示的名称，该名称也称为窗体别名。在创建窗体时，默认的标题和窗体的名称是相同的，修改该属性值能够更改窗体标题栏显示的内容。如，将窗体的标题改为当前工作簿名，可使用下面语句：

```
UserForm1.Caption=Application.Caption
```

在使用用户窗体时，有时需要更改窗体的外观，这包括窗体的大小、背景颜色和使用背景图片等。使用 BackColor 属性和 ForeColor 属性可以设置窗体的背景色和前景色，使用 Height 属性和 Width 属性可以设置窗体的宽度和高度以改变窗体的大小。使用 Picture 属性可以设置窗体中显示的图片，作为窗体的背景图片。如，将窗体的背景色设置为绿色，可使用下面的语句：

```
UserForm1.BackColor = RGB(0, 255, 0)
```

在对窗体编程时，如果需要控制窗体对用户事件是否响应，可以设置窗体的 Enable 属性。当其值设置为 False 时，窗体不响应用户事件，此时窗体不能移动和改变大小，窗体中的控件也不会响应任何操作。该属性的默认值为 True。

提示：用户窗体能使用两种模式来显示，即模式方式和无模式方式。模式窗体，指的是该窗体显示时不允许用户对其他窗体或 Excel 进行其他操作，这意味着窗体独占焦点。无模式窗体指的是不关闭该窗体同样能够执行对其他窗体的操作，也可以操作 Excel 工作表。窗体是模式还是无模式的，可以使用 ShowModal 属性来设置，当其值为 True 时，窗体为模式窗体。在默认状态下，创建的窗体都是模式窗体，当其显示后状态是无法改变的。只有隐藏后再显示，才能指定它为无模式窗体。

窗体与控件一样，同样可以通过窗体事件来响应用户的操作。在使用窗体创建用户界面时，往往需要在窗体加载时进行初始化操作，此时可以通过编写窗体的 Initialize 事件程序来实现。窗体的 Initialize 事件（即用户窗体初始化事件）是在窗体加载触发的事件，因此该事件的事件程序，通常被用来进行对象的初始化操作，如，为公有变量赋初值，为窗体中的“组合框”控件和“列表框”控件添加选项，将窗体中“文本框”控件清空。

在关闭一个窗体时，将会触发 QueryClose 和 Terminate 事件。QueryClose 事件在关闭窗体时前发生，该事件能够给用户一个通过编程取消窗体关闭的机会。Terminate 事件将最终关闭窗体，并且是不能被用户取消的。

提示：在关闭一个用户窗体时，请求关闭（QueryClose）事件将先于终止（Terminate）事件发生。

【范例 15-1】 编写一个窗体程序，程序设置窗体样式，当单击窗体时，窗体会随机改变大小和在屏幕上的位置，同时更改背景色。关闭窗体，程序给出不同事件响应的提示，代码如下所示。

```
Private Sub UserForm_Click()                        '窗体单击事件
    With UserForm1
        .BackColor = RGB(Rnd * 256, Rnd * 256, Rnd * 256)
                                                    '设置窗体背景色
        .Height = Rnd * 680                         '设置窗体高度
        .Width = Rnd * 800                          '设置窗体宽度
        .Top = Rnd * 100                            '设置窗体离屏幕顶端位置
        .Left = Rnd * 100                           '设置窗体离屏幕左侧的位置
    End With
End Sub
Private Sub UserForm_Initialize()                   '窗体初始化事件
    With UserForm1
        .Caption = "跳动的窗体"                      '设置窗体标题
        .Width = 300                                '设置窗体宽度
        .Height = 300                               '设置窗体高度
        .Top = 500                                  '设置窗体离屏幕顶端位置
        .Left = 500                                 '设置窗体离屏幕左侧位置
    End With
End Sub
Private Sub UserForm_QueryClose(Cancel As Integer,
CloseMode As Integer)                               '使用 QueryClose 事件
```

```
21      MsgBox "产生“QueryClose”事件，现在将要关闭窗体！", _
22      vbOKOnly, "发生 QueryClose 事件"          '显示提示信息
23  End Sub
24  Private Sub UserForm_Terminate()          '使用 Terminate 事件
25      MsgBox "窗体马上关闭！",vbOKOnly, _
26      "发生 Terminate 事件"                  '显示提示信息
27  End Sub
```

【运行结果】在工程资源管理器中添加一个窗体，打开窗体的“代码”窗口为窗体添加事件代码。按 F5 键运行程序，窗体初始化的效果如图 15.3 所示。单击鼠标，窗体改变大小和颜色，并随机更改其在屏幕中的位置，如图 15.4 所示。

单击窗体右上角的“关闭”按钮☒关闭窗体，程序将先显示“产生 QueryClose 事件”提示对话框，如图 15.5 所示。然后给出“发生 Terminate 事件”提示对话框，如图 15.6 所示。

【代码解析】本范例演示常见窗体事件的使用。代码第 01 至第 09 行演示 Click 事件过程，其中使用窗体属性设置窗体样式，属性值均为随机数。第 10~17 行代码演示窗体 Initialize 事件过程，在窗体初始化时，设置窗体标题、窗体大小和窗体在屏幕中的位置。第 19~26 行代码是 QueryClose 事件和 Terminate 事件响应程序过程，从演示结果可以看到在关闭窗体时，QueryClose 事件的发生先于 Terminate 事件。

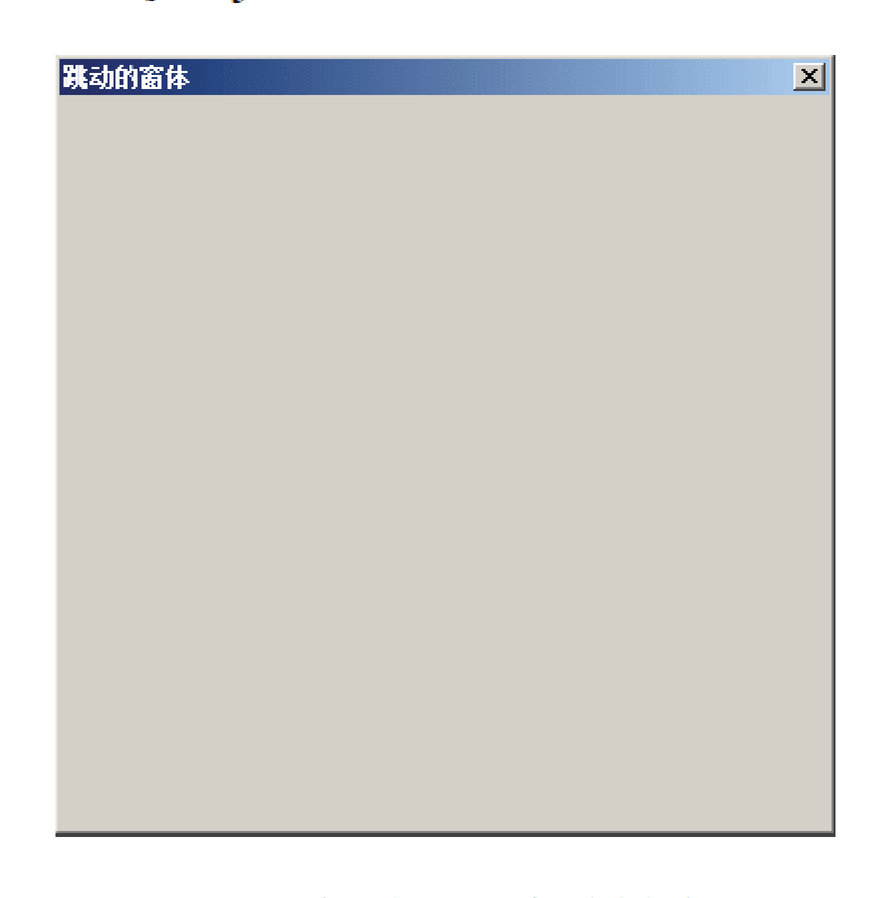
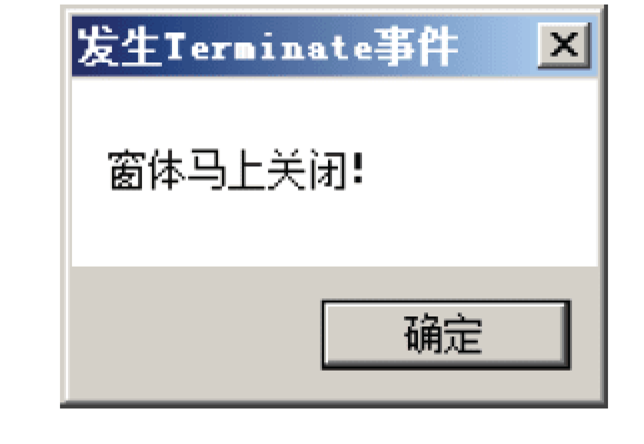


图 15.3　初始化的窗体样式

图 15.4　鼠标在窗体中单击更改背景颜色和窗体大小





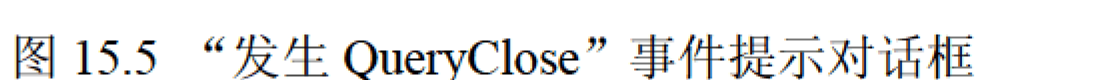


图 15.5　“发生 QueryClose”事件提示对话框

图 15.6　“发生 Terminate 事件”提示对话框

15.1.3　使用窗体的方法

使用窗体的方法可以实现对窗体的各种操作。在程序中，用户窗体往往需要进行各种操作，如改变窗体在屏幕上的位置、使窗体显示或隐藏等。这些操作可以使用窗体的方法

来实现。

用户窗体可以在程序运行时使用 Hide 方法或 Show 方法来控制其是否可见。用户窗体的 Hide 方法可以隐藏一个窗体。窗体隐藏后，用户将不能与应用程序进行交互，直到隐藏结束为止。而使用 Show 方法可以使一个用户窗体对象显示，其参数 modal 用于决定用户窗体是模式的还是无模式的。如果运行 Show 方法时窗体还未加载，则 VBA 会自动加载它。Show 方法的语法格式如下所示：

```
UserForm.Show modal
```

注意：窗体的隐藏并不是将窗体卸载。如果在使用 Hide 方法时窗体还没有装载，则会自动装载窗体，但窗体不会显示。

如果需要改变一个用户窗体在屏幕上的位置，可以使用窗体的 Move 方法来实现。Move 方法可以在屏幕上将窗体移动到需要的位置，同时还能够更改窗体的大小。使用 Move 方法时，其 Left 和 Top 参数指定窗体的位置，使用 Width 和 Height 参数指定窗体的大小，通过 Layout 参数决定移动后控件的父对象是否初始化。该方法的其语法格式如下：

```
Move([Left[,Top,[Width[,Height[,Layout]]]]])
```

【范例 15-2】 编写一个窗体移动动画，使窗体从屏幕的左上角先向右下角移动，在屏幕底部后向屏幕右上角移动，代码如下所示。

```
Private Sub UserForm Click()
    Dim n As Integer                      '声明变量
    Me.Left = 0                           '将窗体放置于屏幕左上角
    Me.Top = 0
    For n = 1 To 1000                     '开始循环
        If n < 500 Then                   '如果 n<500
            Me.Move n, n                  '向右下角移动
        Else
            Me.Move n, 1000 - n           '大于 500 向右上角移动
        End If
    Next
End Sub
```

【运行结果】在工程资源管理器中添加一个窗体，打开窗体的“代码”窗口为窗体添加事件代码。按 F5 键运行程序，单击屏幕上的窗体，窗体放置到屏幕左上角，然后向右下角移动，在屏幕的底部改为向屏幕的右上角移动，如图 15.7 所示。

【代码解析】本范例使用窗体的 Move 方法来实现窗体的移动。程序通过窗体的 Click 事件触发，在 For...Next 循环结构中使用 Move 方法，不断地在屏幕上移动窗体获得动画效果。为了实现窗体运动方向的改变，使用 If 语句来对 n 的值进行判断，当 n 的值小于 500，Move 方法的 Left 和 Top 参数均为 n 值，不断增大，则窗体向右下角移动。当 n 的值大于 500 后，Move 方法的 Left 值随着 n 的增大而增大，窗体仍向右移动；而 Top 减小，窗体向上移动，这样获得窗体向右上角运动的效果。

注意：Me 是一个在用户窗体代码中指定当前用户窗体的快捷方式，这个关键字也适用于当前控件的引用。在使用 Move 方法时，如果使用命名参数，则参数可以任意

摆放。否则，参数必须按照语法格式的顺序来放置。没有指定的参数将保持其原有的值不变。

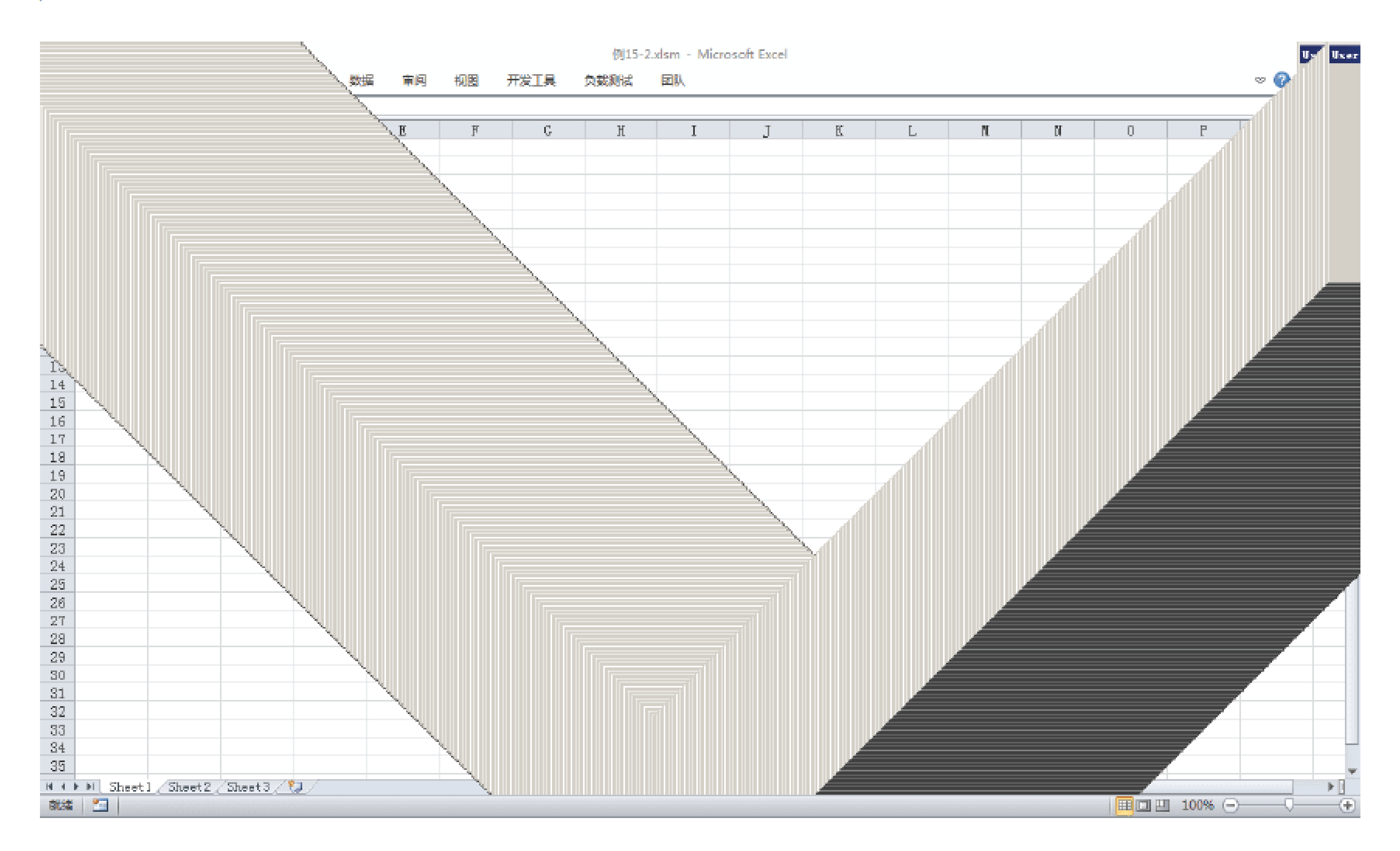


图 15.7　窗体在屏幕上移动

15.2　使用 Excel 控件

设计用户界面的过程，实际上就是向窗体中添加控件的过程。窗体是载体，而控件是置于载体中用来实现各种功能的工具。窗体只有被赋予了控件，才能形成真正意义上的用户操作界面。本节将首先介绍 VBA 控件工具箱中标准控件的使用方法。

15.2.1　认识控件

控件工具箱是控件的仓库，其为程序设计者提供了可选择的控件，向窗体添加控件的过程，实际上就是将控件工具箱中的控件放置到用户窗体中的过程。在 VBA 中，当插入一个用户窗体后，Visual Basic 编辑器会自动打开控件工具箱，如图 15.8 所示。用户只需要将其中的控件按钮拖放到用户窗体中即可实现控件的添加。



图 15.8　控件工具箱

控件工具箱直接提供了 15 种控件，分别是“标签”控件、“文本框”控件、“组合框”控件、“列表框”控件、“复选框”控件、“单选按钮”控件、“切换按钮”控件、“框架”控件、“命令按钮”控件、“TabScrip”控件、“多页”控件、“滚动条”控件、“旋转按钮”控件、“图像”控件和 Refedit 控件。在

工具箱中单击这些控件按钮选中控件后，即可通过在窗体中拖动鼠标将其添加到窗体中了。

注意：在控件工具箱中的这些控件中，前面 7 个控件、“命令按钮”控件以及“滚动条”控件和“图像”控件在 Excel 2010 的“开发工具箱”选项卡的“插入”列表中的“ActiveX 控件”栏中都能够找到。这里的“框架”控件与上一章提到的“分组”控件作用相同，“旋转按钮”控件与前面介绍的“数值调节钮”控件功能相同。

除了控件工具箱上的控件之外，在窗体上还可以添加附加控件。要向窗体中添加附加控件，可使用下面步骤来进行操作：

（1）在控件工具箱中右击，在弹出的快捷菜单中选择“附加控件”命令，打开“附加控件”对话框。在“可用控件”列表中选择需要添加的控件，如图 15.9 所示。

（2）单击“确定”按钮关闭对话框，选择的控件即被添加到控件工具箱中。选择该控件，在窗体中拖动光标，即可将控件添加到窗体中，如图 15.10 所示。

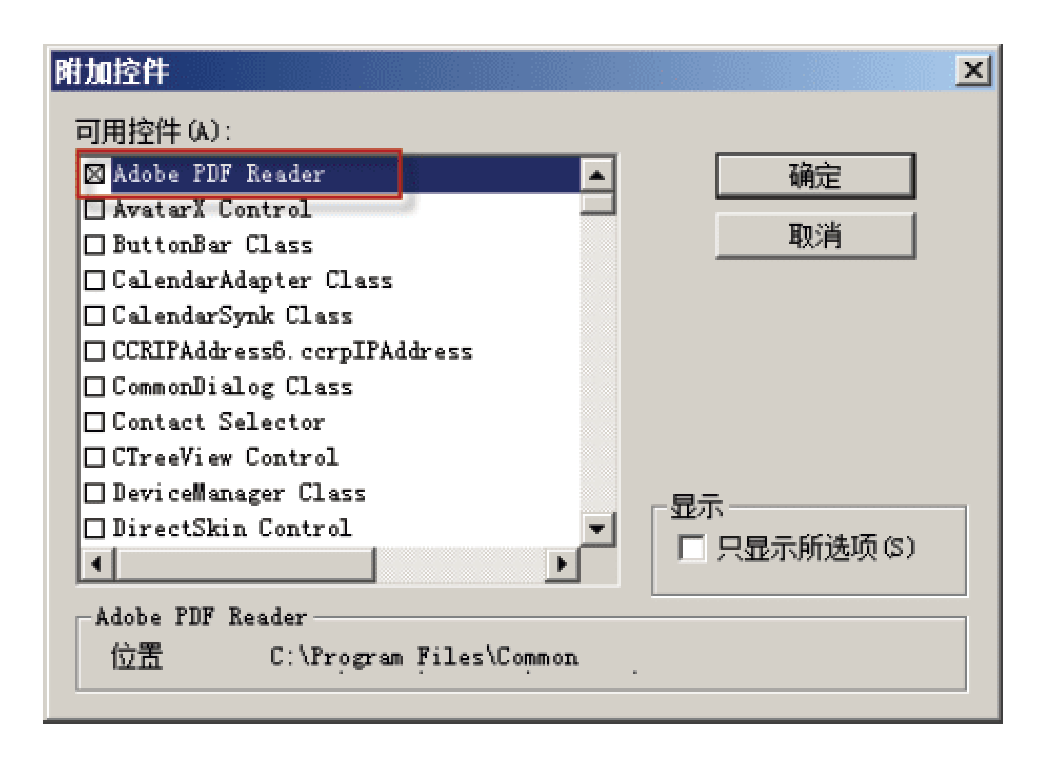


图 15.9　“附加控件”对话框

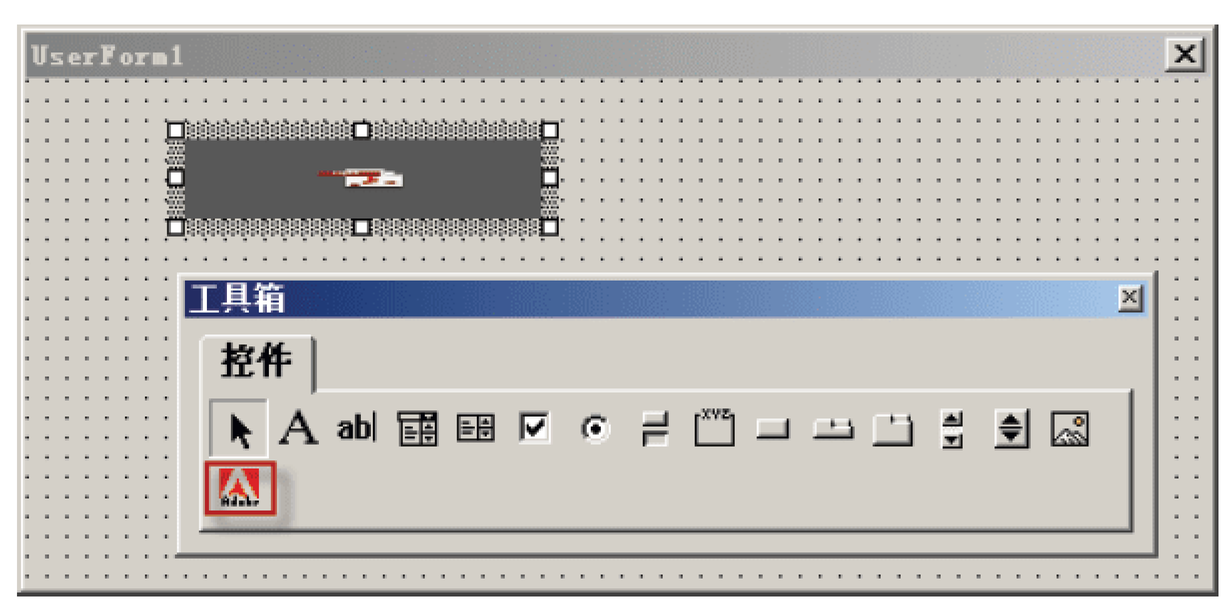


图 15.10　向窗体添加控件

15.2.2　设置窗体控件

在窗体中添加完成控件后往往需要对其进行设置，如调整其大小、设置控件在窗体上的布局以及设置控件的 Tab 键序等。在下面的实例中，窗体中放置了 3 个按钮控件，下面将调整这 3 个控件在窗体中的布局，使它们水平等距排列。其具体的操作步骤如下：

（1）设置控件大小。在窗体中的控件可以通过拖动控件边框上的 8 个控制柄来改变控件的大小。如果要使这些控件同时获得相同的大小尺寸，使用光标拖动的方法就很难做到。此时可以采用下面的方法来设置。按住 Shift 键，单击需要调整大小的控件同时选择它们，在任意一个控件上右击，在弹出的快捷菜单中选择“统一尺寸”→“两者都相同”命令，如图 15.11 所示。此时，窗体中选择控件的大小将变得相同，如图 15.12 所示。

提示：按 Ctrl 键，依次单击需要选择的控件，也可以将它们同时选择。同时选择多个控件后，拖动一个控件边框上的控制柄调整其大小，其他控件的大小也会随之发生改变。

（2）对齐放置控件。添加在窗体中的控件，在排列上有时是杂乱的，需要让它们对齐放置。此时，可以同时选择需要对齐放置的控件后右击，在弹出的快捷菜单中选择“对齐”命令的下级菜单命令，即可执行控件的对齐操作。如，这里选择“中间对齐”命令，如图

15.13 所示。此时，被选中的控件将按中心为基准对齐，如图 15.14 所示。

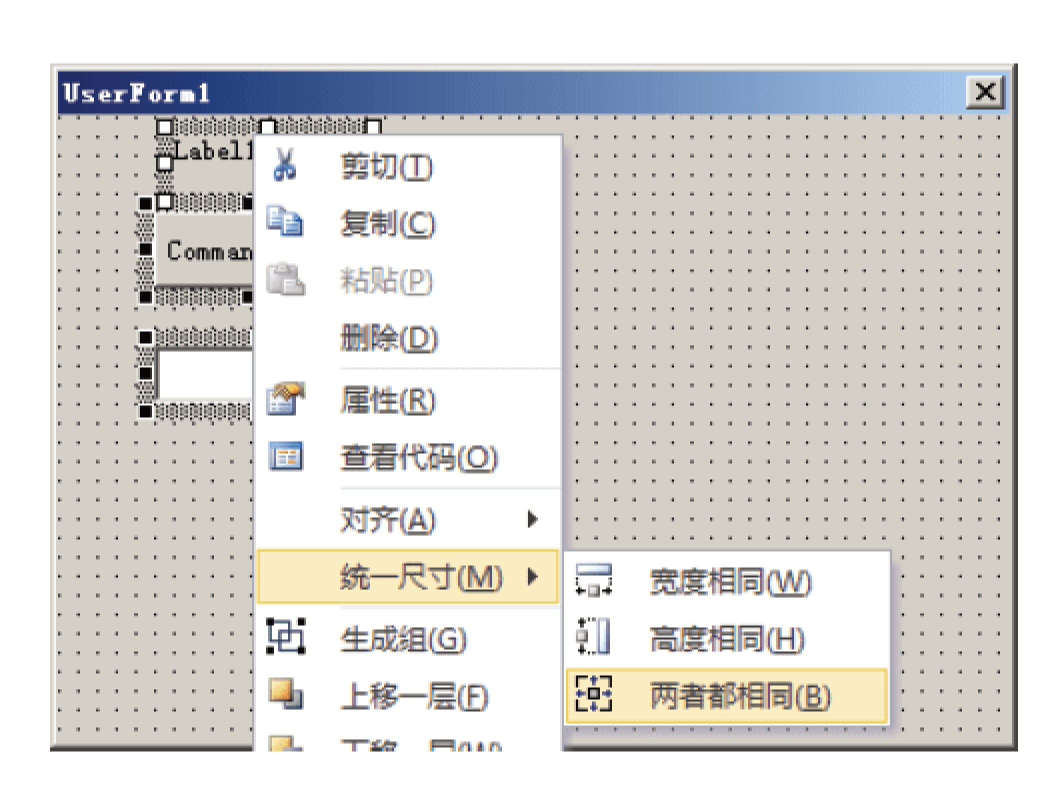


图 15.11　选择“两者都相同”命令

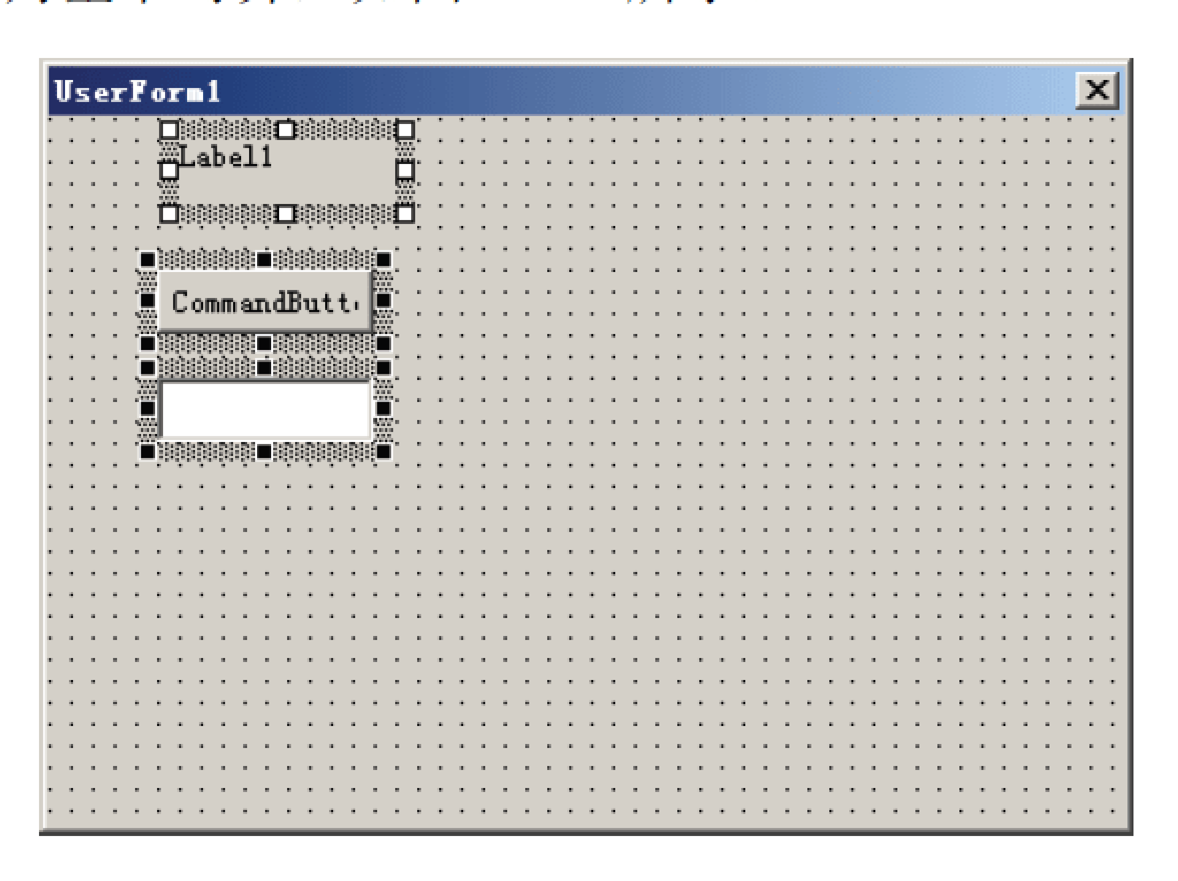


图 15.12　控件大小变得相同

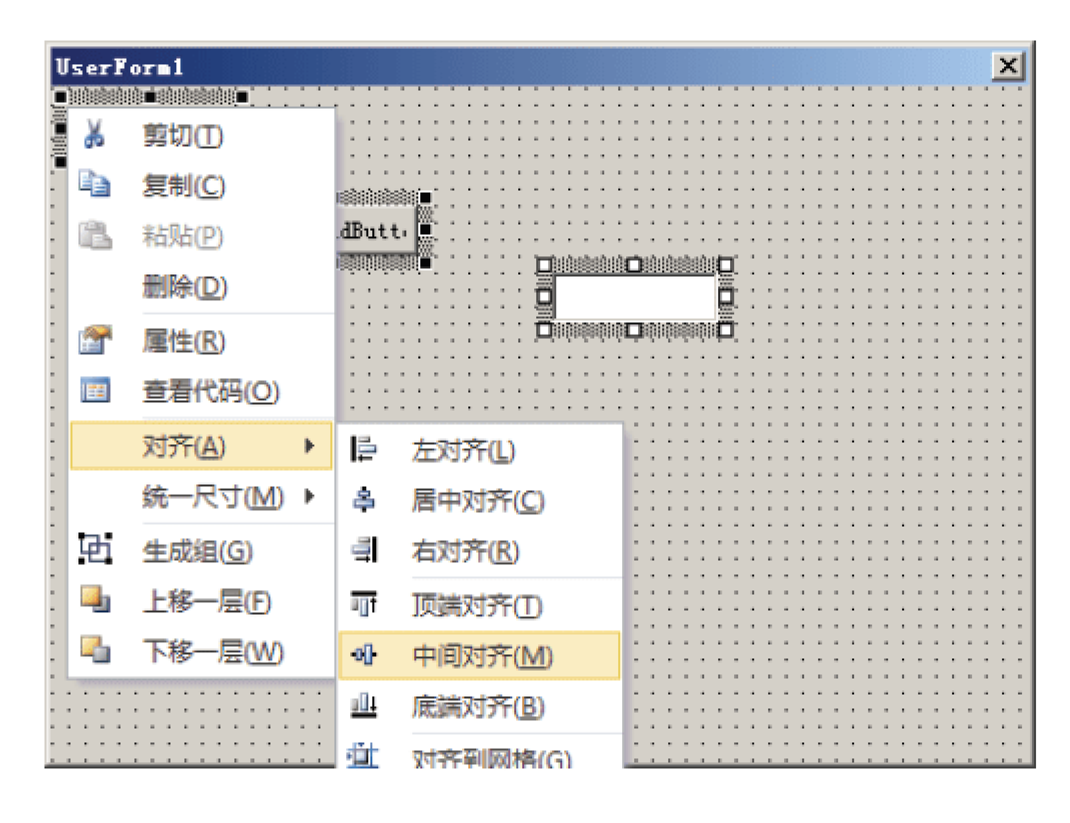


图 15.13　选择“中间对齐”命令

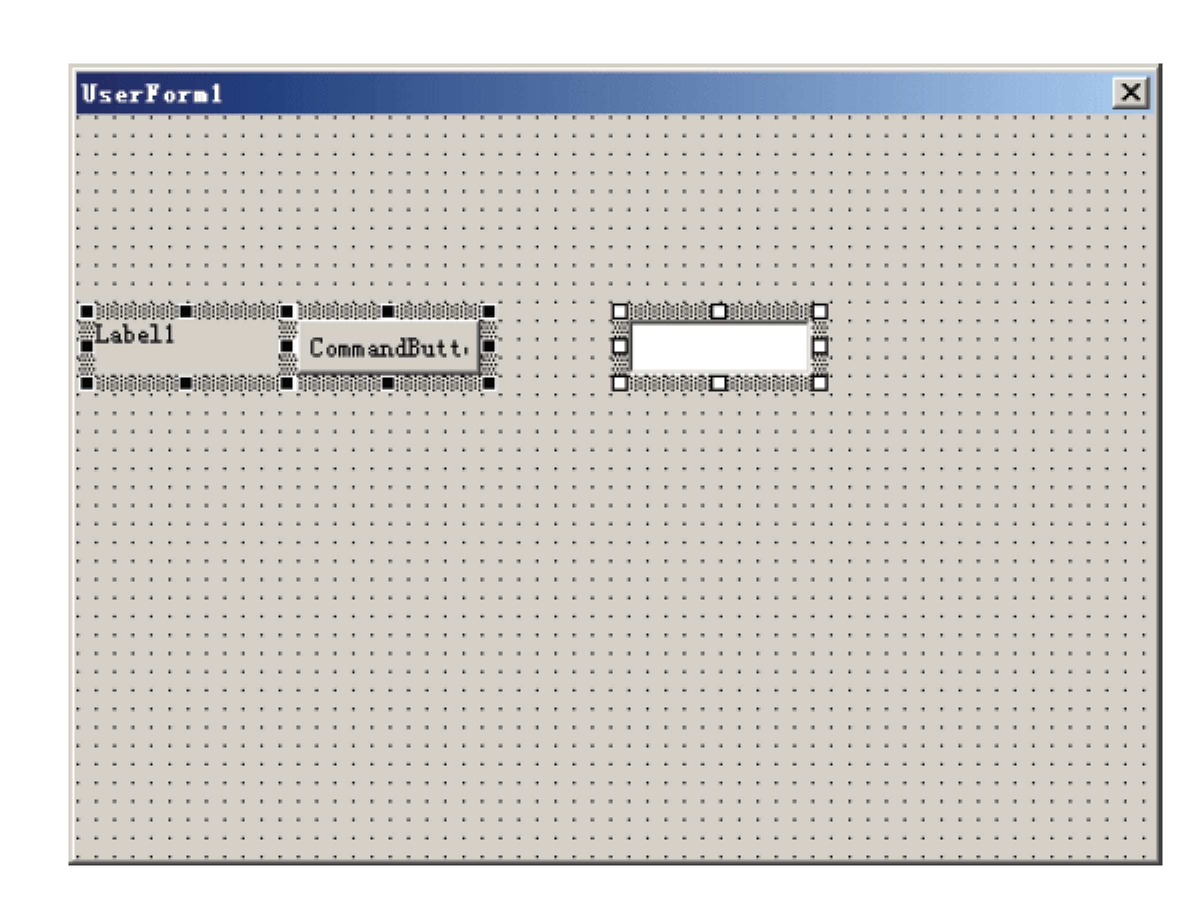


图 15.14　控件中间对齐的效果

（3）均匀分布控件。这里，控件的分布还不均匀，可以在水平方向上使控件均匀分布。选择“格式”→“水平间距”→“相同”命令，则选择的对象将在水平方向上均匀分布，如图 15.15 所示。

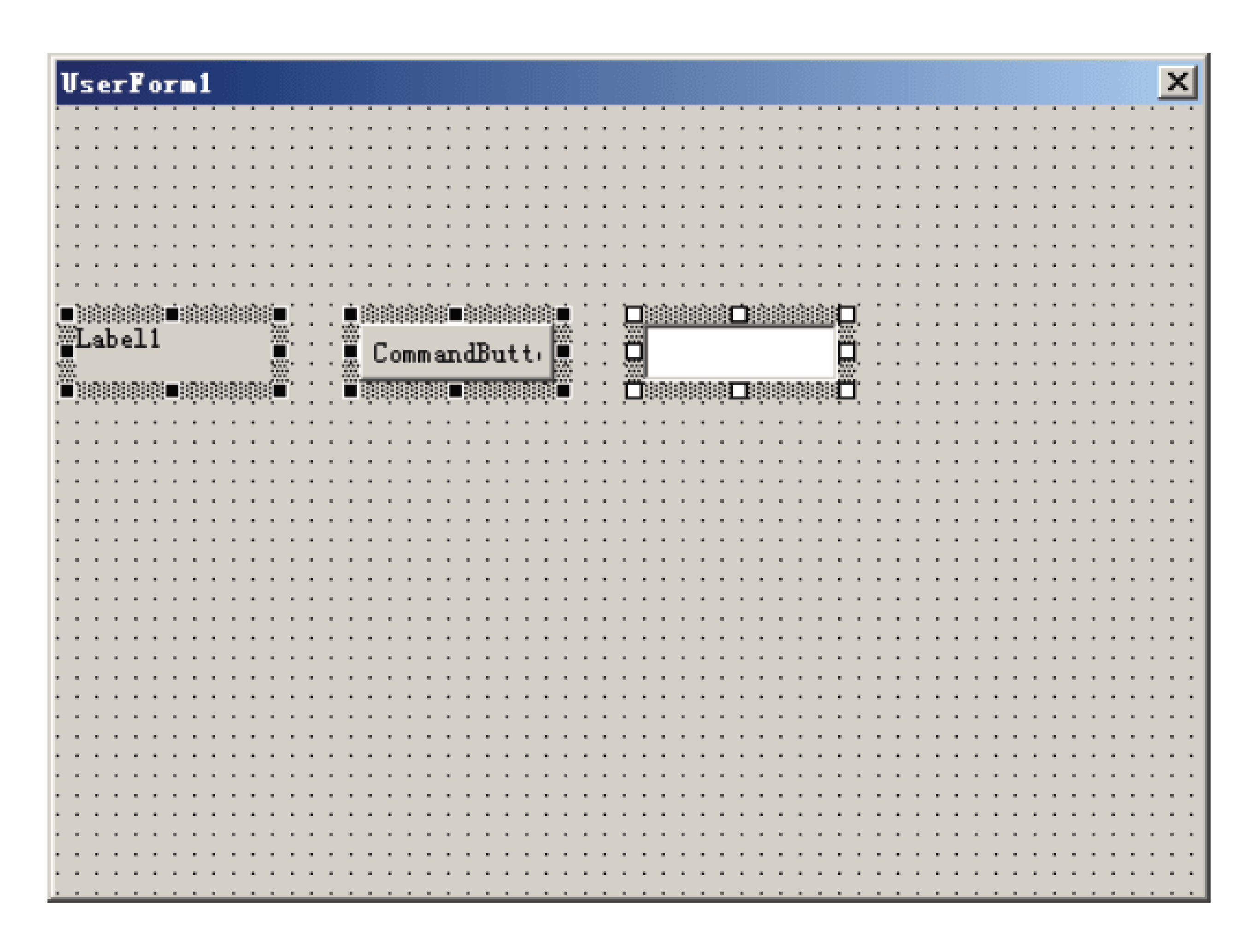


图 15.15　控件在水平方向上均匀分布

提示：在“格式”→“水平间距”命令的下级菜单中，还包括“递增”、“递减”和“移除”命令，可实现按照间距递增、递减和边框相连的方式来排列控件。同样，如果需要在垂直方向上排列控件，可以选择“格式”→“垂直间距”菜单命令来实现。

（4）调整控件层次关系。在窗体中添加一个“图像”控件，该控件将盖住前面添加的“按钮”控件，如图 15.16 所示。这是因为在窗体中，后添加的控件位于先添加的控件的上方。选择“格式”→“顺序”命令，在其下级菜单中包括“移至顶层”，“移至底层”、“上移一层”和“下移一层”这 4 个命令。选择这 4 个命令中需要的命令，即可实现对控件层次关系的修改。如，这里选择“移至底层”命令，将“图像”控件移至最底层，先于它的控件将不会再被遮盖，如图 15.17 所示。

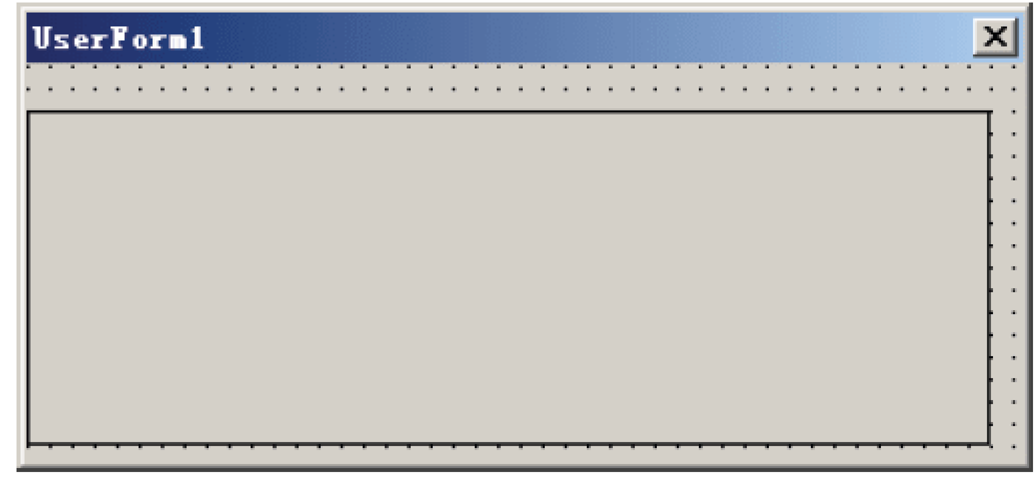


图 15.16　新控件盖住原来的控件

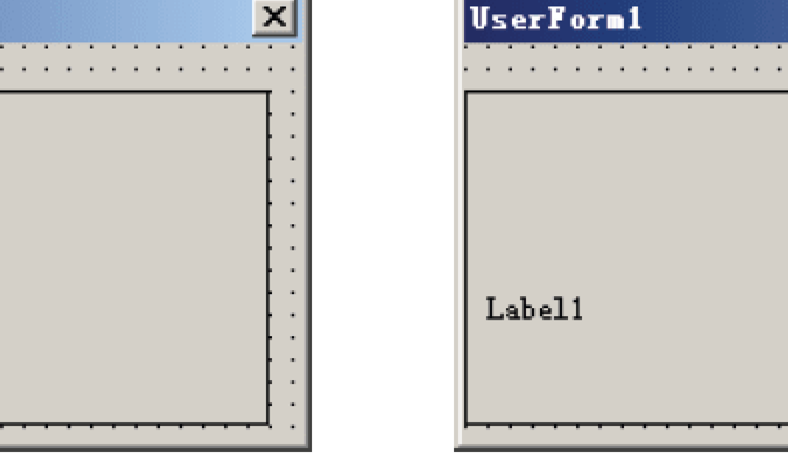


图 15.17　移至底层后的效果

（5）设置 Tab 键顺序。在应用程序中，使用 Enter 键相当于单击处于焦点的控件，按 Tab 键可以改变控件的焦点。按 Tab 键时，控件焦点改变的顺序可以在 Visual Basic 编辑器中进行设置。选择“视图”→“Tab 键顺序”命令，打开“Tab 键顺序”对话框，此时可以在“Tab 键顺序”列表中看到默认的 Tab 键顺序，如图 15.18 所示。在选择某个控件后，单击“上移”按钮或“下移”按钮，改变其在列表中的位置即可改变 Tab 顺序，如图 15.19 所示。





图 15.18　默认的 Tab 键顺序

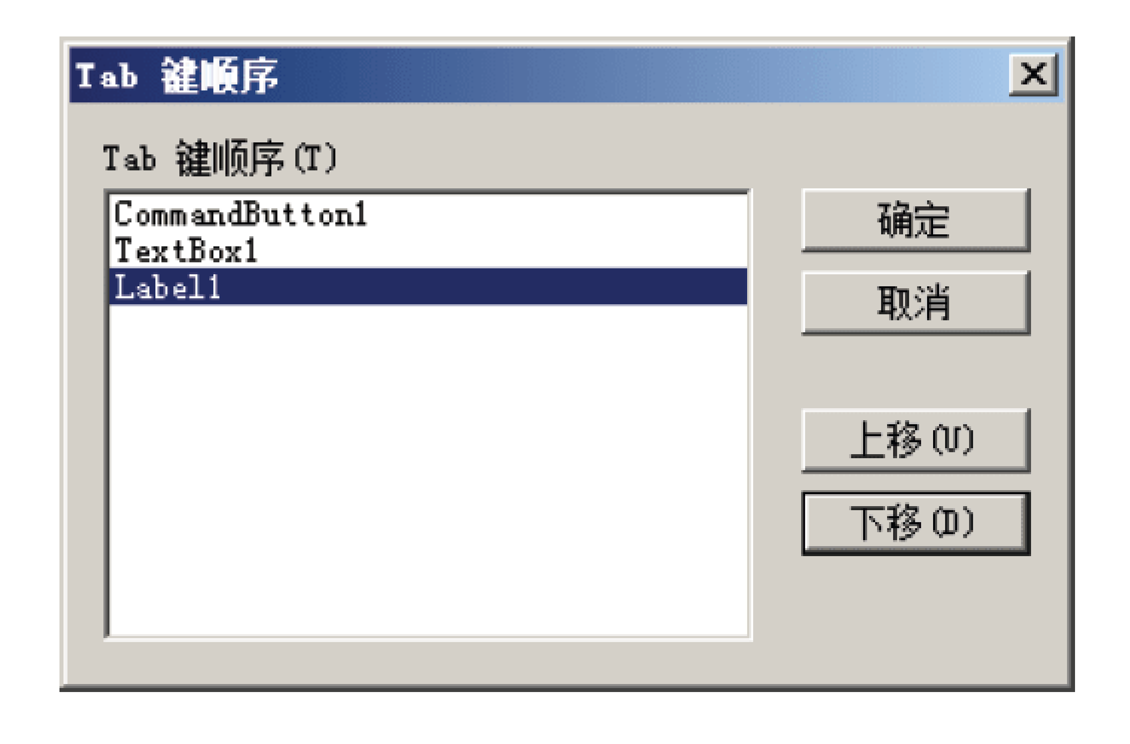


图 15.19　改变控件的 Tab 顺序

提示：焦点指的是对象具有接收键盘或鼠标输入的能力。当对象具有焦点时，就拥有了接受用户输入的能力。在 Windows 环境中，同一时间内只能有一个窗体或控件具有焦点。

15.3　使用 Excel 标准控件

控件是窗体的基本构成元素，VBA 中能够使用的控件包括标准控件和附加控件。标准控件指的是控件工具箱中的 15 个基本控件，这些控件能够直接在工具箱中选择使用。本节将对常用的标准 ActiveX 控件的使用进行介绍。

15.3.1 “标签”控件

“标签”控件是窗体上的一种常见控件，常用来显示固定的提示信息，通常使用该控件作为文本框、列表框和组合框等控件的附加描述信息。

使用“标签”控件时，控件中显示文字的样式往往需要进行设置，控件的 Font 属性可以用来设置“标签”控件显示的字体。该控件可以在“属性”对话框中进行设置，也可以在程序中进行设置。在“属性”对话框中可以设置该属性，如图 15.20 所示。

在程序中，修改“标签”控件的 TextAlign 属性值，可以设置控件中文本对齐方式。这里的文本对齐方式有三种，当设置为 fmTextAlignLeft 时，文本第一个字符将在控件显示区中左对齐；当设置为 fmTextAlignCenter 时，文本在控件显示区中居中对齐；当设置为 fmTextAlignRight 时，文本最后一个字符在控件显示区中右对齐。



图 15.20　设置控件的 Font 属性

【范例 15-3】 编写程序，设置“标签”控件的外观，代码如下所示。

```
Private Sub UserForm Initialize()
    With Label1.Font
    .Bold = True                                          '文字加粗
        .Italic = True                                    '文字斜体
        .Name = "宋体"                                    '设置字体
        .Size = 10                                        '设置文字大小
    End With
    With Label1
        .Caption = "标签控件的常用属性：" & Chr(13) _
        & "Acclerator: " & "用于指定快捷键。" & Chr(13) _
        & "BorderStyle: " & "用于设置边框样式。" & Chr(13) _
```

```
12             & "Font: " & "用于设置标签文字样式。"               '设置标签文字
13             .AutoSize = True                                  '自动改变控件大小
14             .BorderStyle = fmBorderStyleSingle                '设置边框
15         End With
16     End Sub
17     Private Sub Label1_Click()
18         Label1.Font.Name = "隶书"                               '更改字体
19     End Sub
20     Private Sub Label1_DblClick(ByVal Cancel As MSForms.ReturnBoolean)
21         Label1.Visible = False                                 '使控件不可见
22     End Sub
```

【运行结果】在工程资源管理器中添加一个用户窗体，在窗体中添加一个“标签”控件。打开用户窗体的“代码”窗口，输入上述代码。按 F5 键运行程序，窗体中控件的显示效果，如图 15.21 所示。单击“标签”控件，文字字体改变，如图 15.22 所示。双击“标签”控件，控件不可见，如图 15.23 所示。



图 15.21　窗体初始化后控件显示效果



图 15.22　单击控件改变字体



图 15.23　双击控件后控件不可见

【代码解析】本范例演示“标签”控件的属性设置方法。代码的第 02~07 行使用 With 结构设置控件标签文字的属性。第 08 行～15 行代码设置标签显示文字和标签本身的一些属性。控件属性的设置，放在窗体的初始化过程中进行。“标签”控件具有常见的事件，如鼠标单击（Click）事件和鼠标双击（DbClick）事件，范例代码中分别使用这两个事件来实现控件字体的修改和控件的隐藏。

15.3.2　“文本框”控件

“文本框”控件常用来显示或输入各种信息。使用“文本框”控件，用户可以实现文本的输入，即控件作为一个文本输入工具来使用。同时，用户也可以使用该控件来显示大段的文字信息。

“文本框”控件可以用来显示文字信息，文字的内容由控件的 Text 属性值决定。该属性值可以在“属性”对话框中设置，也可以在程序代码中设置。当控件中需要显示的文字

较多时，可以将 MultiLine 属性设置为 True，使文字多行显示，否则文字将只会单行显示。如，使文本框显示“我的文稿”，可使用下面的语句：

```
TextBox1.Text="我的文稿"
```

在使用文本框输入文字时，有时用户并不希望输入的文字在文本框中显示出来，此时通过修改 PasswordChar 属性值来设置替代输入字符的符号。如，当将该属性值设置为“*”时，文本框中输入字符将显示为“*”。这个属性对于将文本框作为密码输入框时十分有用，可以使用下面语句来实现这一功能：

```
TextBox1.PasswordChar="*"
```

提示： 使用“文本框”控件往往需要根据控件输入的状态来编写相应的处理程序，其常用的事件为 Change 事件，该事件在文本框中内容发生改变时触发。编写该事件的事件程序，可以产生文本框内容改变后的动作。

下面以制作一个登录界面为例，来进一步介绍属性“文本框”控件的使用方法。程序运行时，出现用户登录窗体，用户在窗体的文本框中输入用户名和密码正确，则显示欢迎窗体，否则显示提示信息。

（1）启动 Excel 2010，按“Alt+F11”快捷键，打开 Visual Basic 编辑器。在工程资源管理器中右击，在弹出的快捷菜单中选择“插入”→“用户窗体”命令，插入一个用户窗体。在用户窗体中插入两个“标签”控件、两个“文本框”控件和两个“命令按钮”控件。在用户窗体中调整这些控件的位置和大小，如图 15.24 所示。

（2）右击“标签”控件，在弹出的快捷菜单中选择“属性”命令，打开“属性”对话框。在“属性”对话框中更改控件的 Caption 属性值。这里将两个控件的 Caption 属性设置为需要的提示文字，如图 15.25 所示。

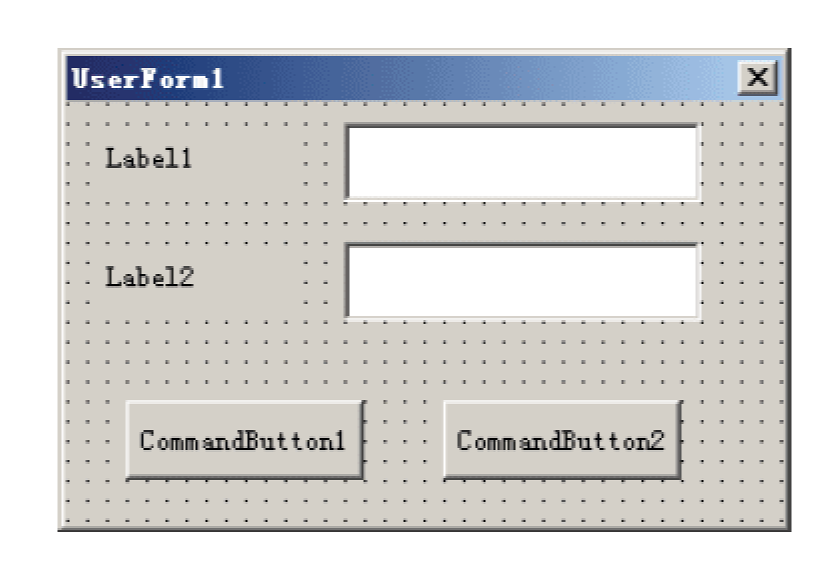



图 15.24　向用户窗体中添加控件

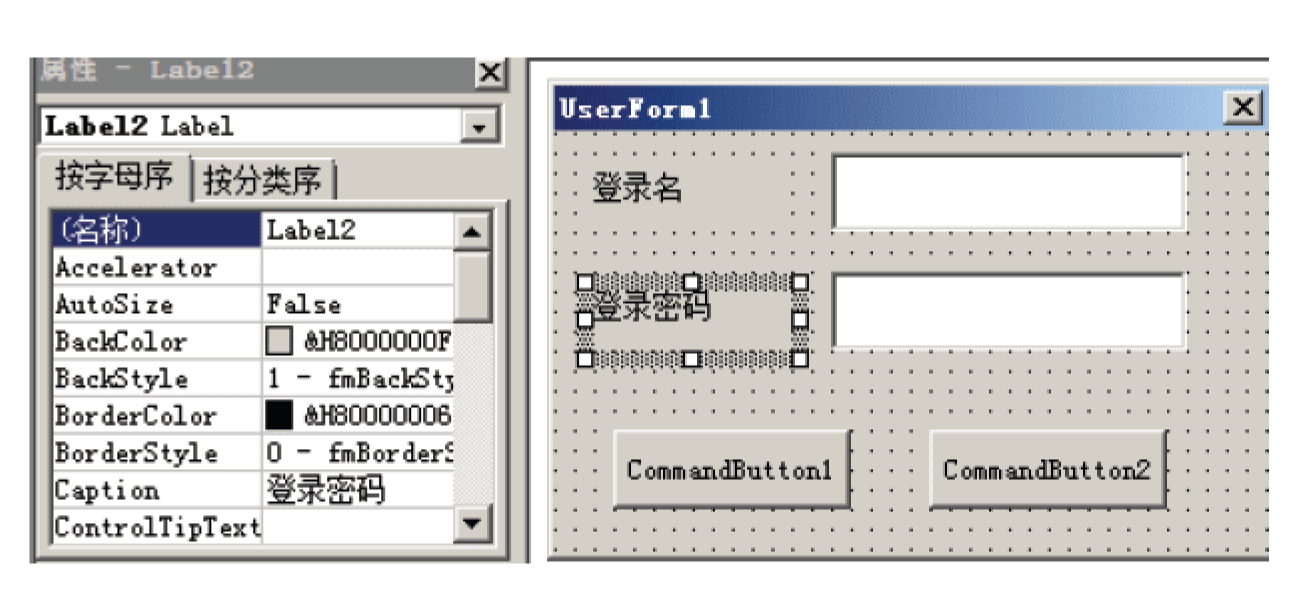


图 15.25　更改“标签”控件的属性

（3）使用相同的方法更改用户窗体和两个“命令按钮”控件的 Caption 属性，更改后的效果如图 15.26 所示。

（4）选择第二个“文本框”控件，在“属性”对话框中将 PasswordChar 属性设置为字符“*”，如图 15.27 所示。

（5）制作一个欢迎窗体。这里首先插入一个用户窗体，更改其窗体名称。在窗体中添加一个“标签”控件，输入提示文字并设置文字的样式，如图 15.28 所示。

（6）右击“确定”按钮，在弹出的快捷菜单中选择“查看代码”命令，打开“代码”窗口。Visual Basic 编辑器自动在“代码”窗口中添加一个按钮的 Click 事件，如图 15.29

所示。

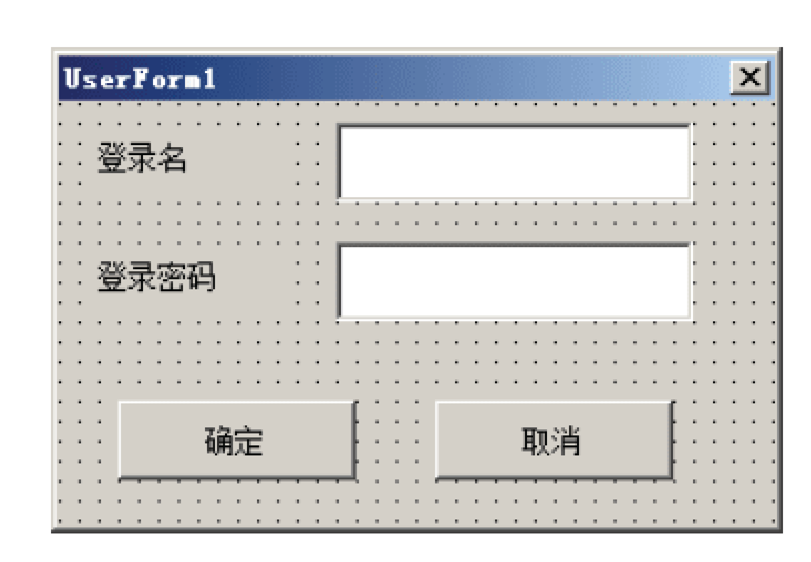


图 15.26　更改其他控件 Caption 属性后的效果

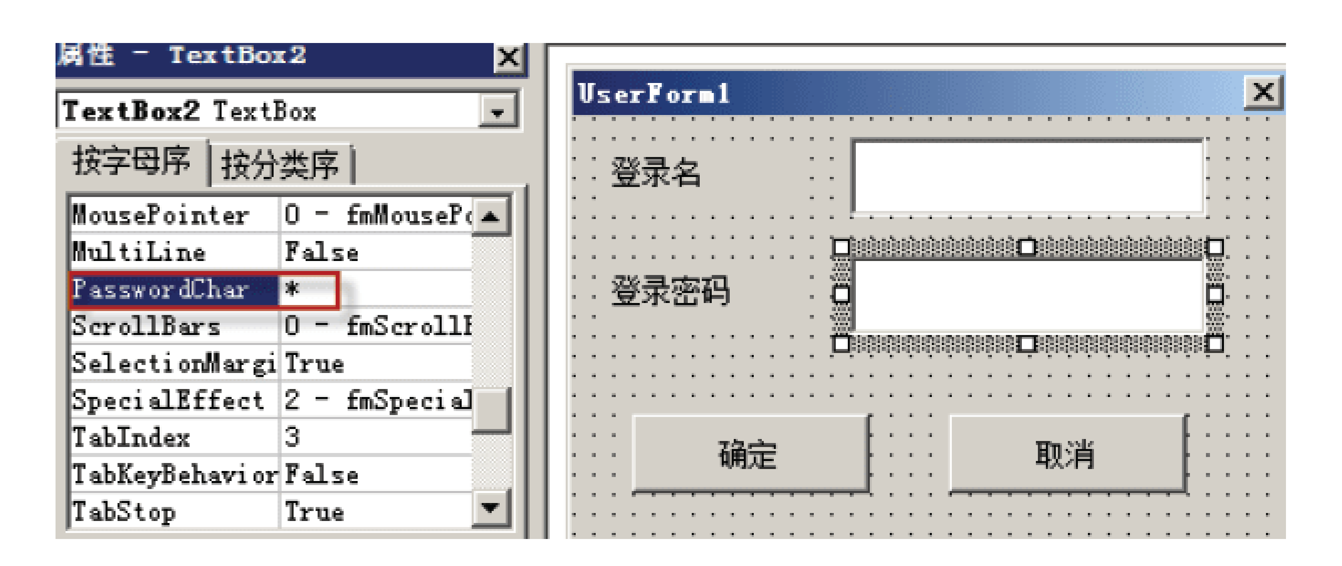


图 15.27　设置 PasswordChar 属性

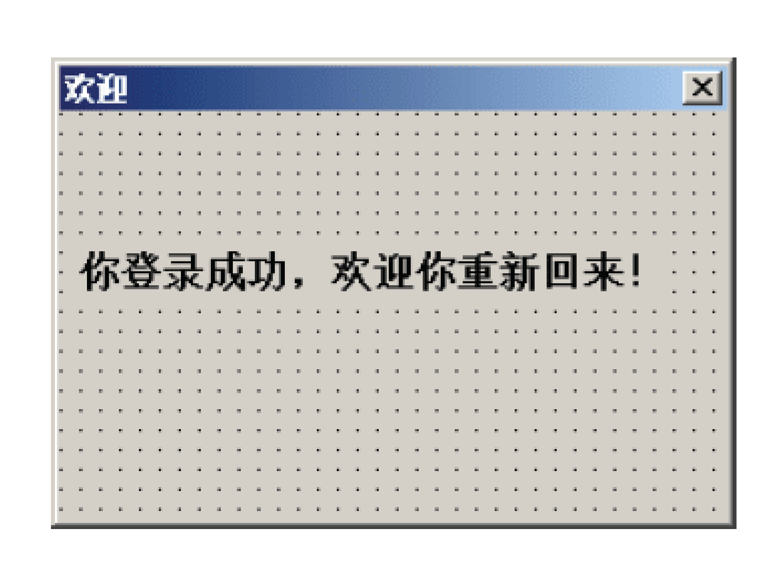


图 15.28　创建第二个用户窗体

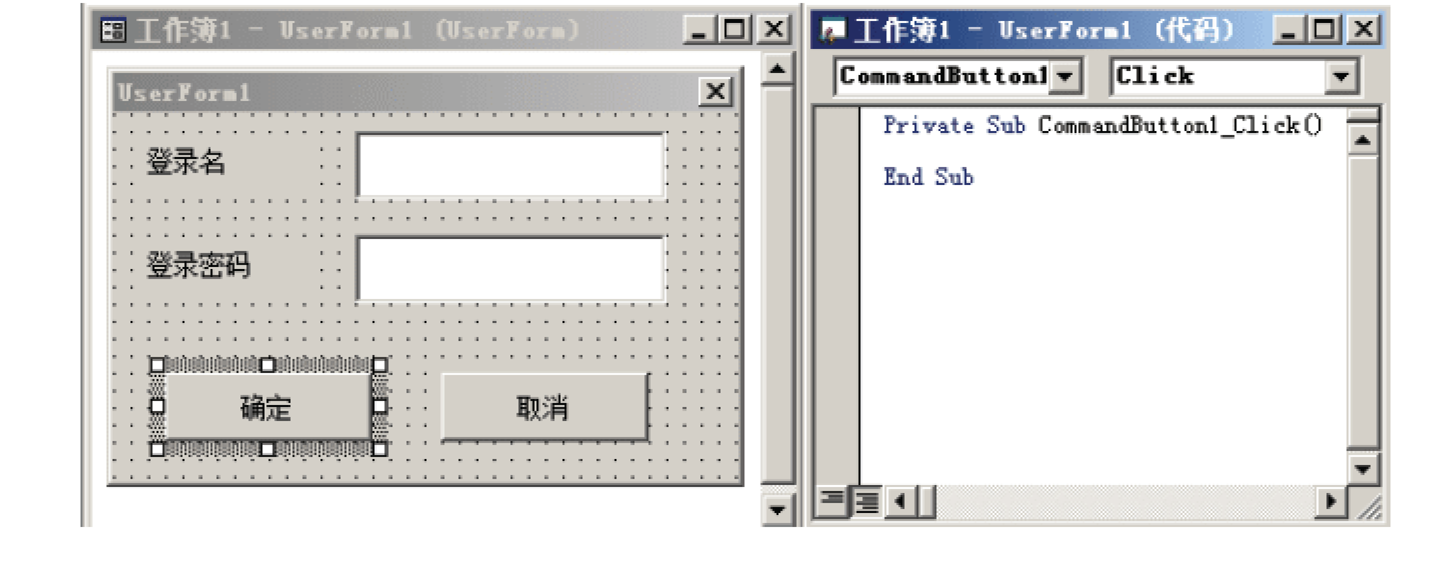


图 15.29　添加按钮 Click 事件

（7）在“代码”窗口中为“确定”按钮添加如下的事件代码：

```
Private Sub CommandButton1_Click()
    If TextBox1.Text <> "郭轶凡" Then                  '判断用户名是否正确
        MsgBox "用户登录名不对，您无权登录！"            '不正确给出提示
        With TextBox1                                  '设置第一个文本框属性
            .SelStart = 0                              '设置选择文本的开始字符
            .SelLength = Len(TextBox1.Text)            '设置选择文本的长度
            .SetFocus                                  '文本框获得焦点
        End With
    ElseIf TextBox2.Text <> "abcdef" Then              '如果密码错误
        MsgBox "密码输入错误，请重新输入！"              '给出提示
        With TextBox2                                  '设置第二个文本框属性
            .SelStart = 0                              '设置选择文本的开始字符
            .SelLength = Len(TextBox1.Text)            '设置选择文本的长度
            .SetFocus                                  '获得焦点
        End With
    Else
        UserForm2.Show                                 '登录成功，显示欢迎窗体
    End If
End Sub
```

提示： 在“确定”按钮的 Click 事件代码中，首先判断第一个文本框的 Text 属性值是否是设定的用户名。如果不是，给出提示，同时使用 With 结构设置 SelStart 属性和 SelLength 的值，使文本框中文字高亮显示。使用 SetFocus 方法使文本框获得焦点，使用户能够直接修改用户名。如果输入的是设定的用户名，则检验第二个文本框的 Text 属性值是否为设定的密码，如果不是正确密码，执行与用户名错误相同的操作。如果用户名后密码输入均正确，则显示第二个用户窗体。

（8）为“取消”按钮添加如下的事件代码：

```
Private Sub CommandButton2_Click()
    Unload Me                                          '卸载窗体
End Sub
```

（9）按 F5 键运行程序，程序给出“用户登录”窗口，要求输入登录名和密码。在窗口中输入登录名和密码，如图 15.30 所示。如果登录名和密码全部输入正确，单击“确定”按钮后，将显示登录成功的欢迎提示，如图 15.31 所示。如果用户名输入错误，程序将先给出提示，如图 15.32 所示。关闭提示对话框后，错误的用户名将高亮显示，用户可以直接输入新的用户名，如图 15.33 所示。如果密码输入错误，程序的运行效果与用户名错误时一样。在任何时候单击“取消”按钮，窗体将关闭。



图 15.30　输入登录名和密码

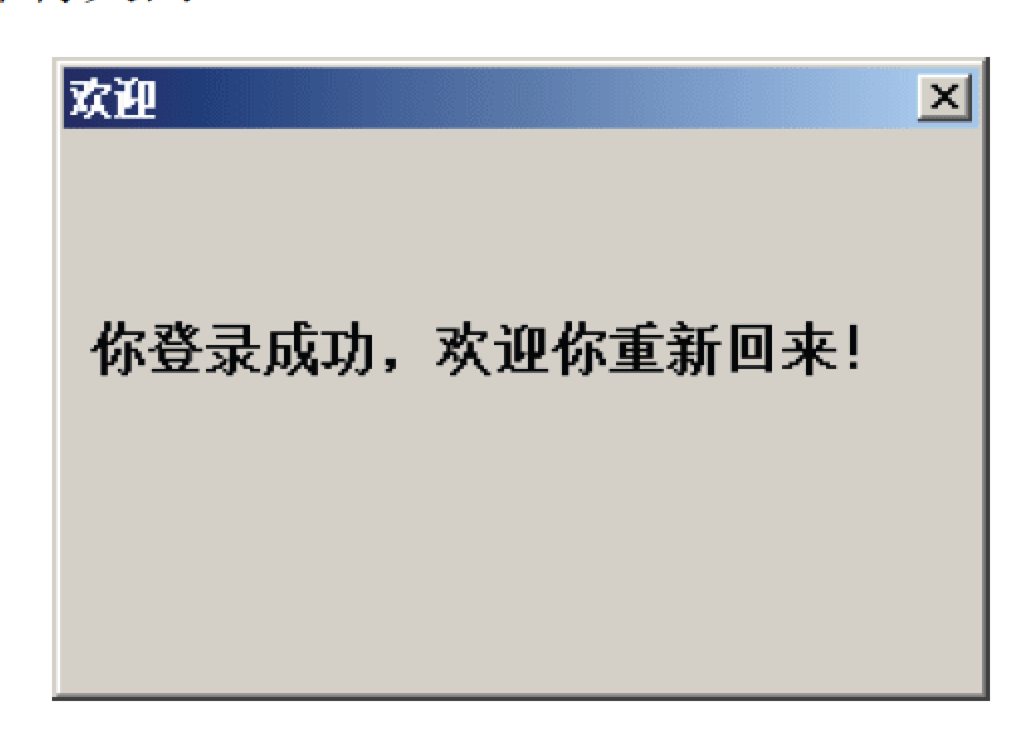


图 15.31　登录成功的欢迎提示

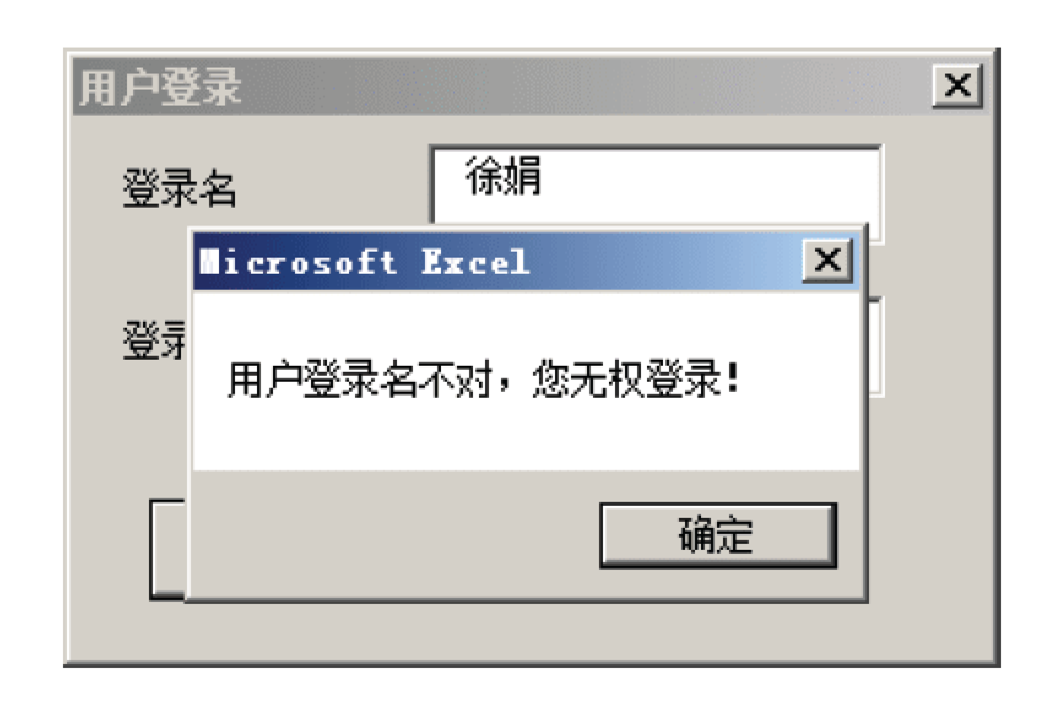


图 15.32　登录名输入错误时的提示



图 15.33　“登录名”输入框中文字高亮显示

15.3.3　“复选框”控件和“单选按钮”控件

“复选框”控件和“单选按钮”控件都是用来对状态进行选择的控件，它们的区别仅仅在于“复选框”控件是独立的互不影响的，而一组“单选按钮”控件的状态是相互关联的。在程序中“复选框”控件可以单独使用，而“单选按钮”控件常常成组使用。“复选框”控件在应用程序中常用来对特定状态进行选定或清除，“单选按钮”控件常用于在一组方案中选择一种。

“复选框”控件和“单选按钮”控件拥有大多数控件所具有的属性，下面以一个实例

来介绍这两种控件的使用方法。本实例将使用“复选框”控件来控制窗体是否显示背景图片，使用“单选按钮”控件来实现窗体背景图片的选择。

（1）打开 Excel 2010，再打开 Visual Basic 编辑器。在工程资源管理器中添加一个用户窗体，设置窗体的属性，如图 15.34 所示。

提示：这里，将 BackColor 属性设置为黑色，设置 Caption 属性值更改窗体名称。由于窗体将添加背景图片，将 PictureSizeMode 属性设置为 fmPictureSizeModeStretch，添加的图片将被拉伸占满整个窗体。

（2）在窗体中放置一个“标签”控件和一个“文本框”控件，设置“标签”控件的 Caption 属性和 ForeColor 属性，如图 15.35 所示。同时，将控件的 BackStyle 属性设置为 fmBackStyleTransparent，如图 15.36 所示。

（3）在用户窗体中有 1 个“复选框”控件，分别设置控件的 fmBackStyleTransparent 属性、Caption 属性和 ForeColor 属性，如图 15.37 所示。

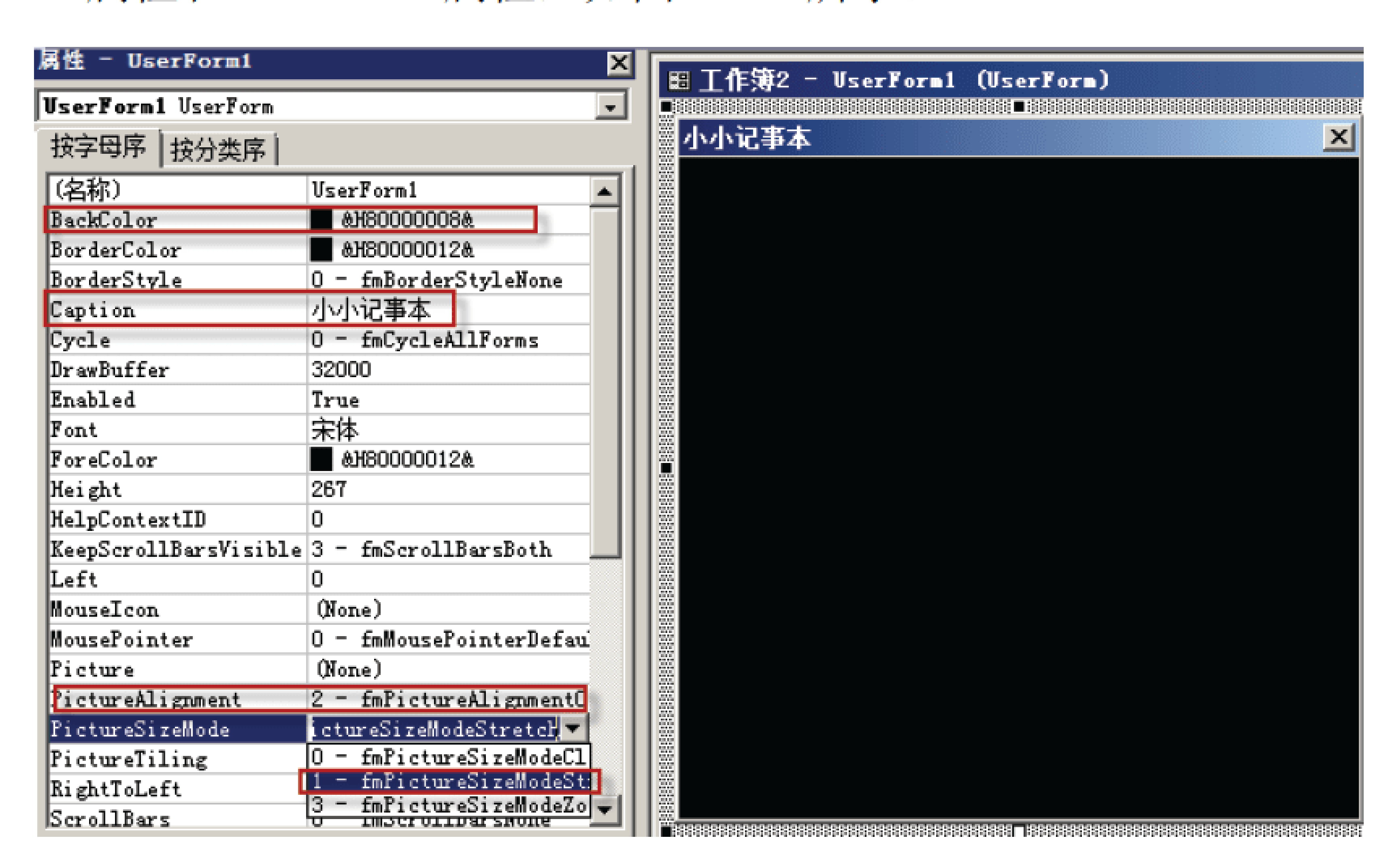


图 15.34　设置窗体属性

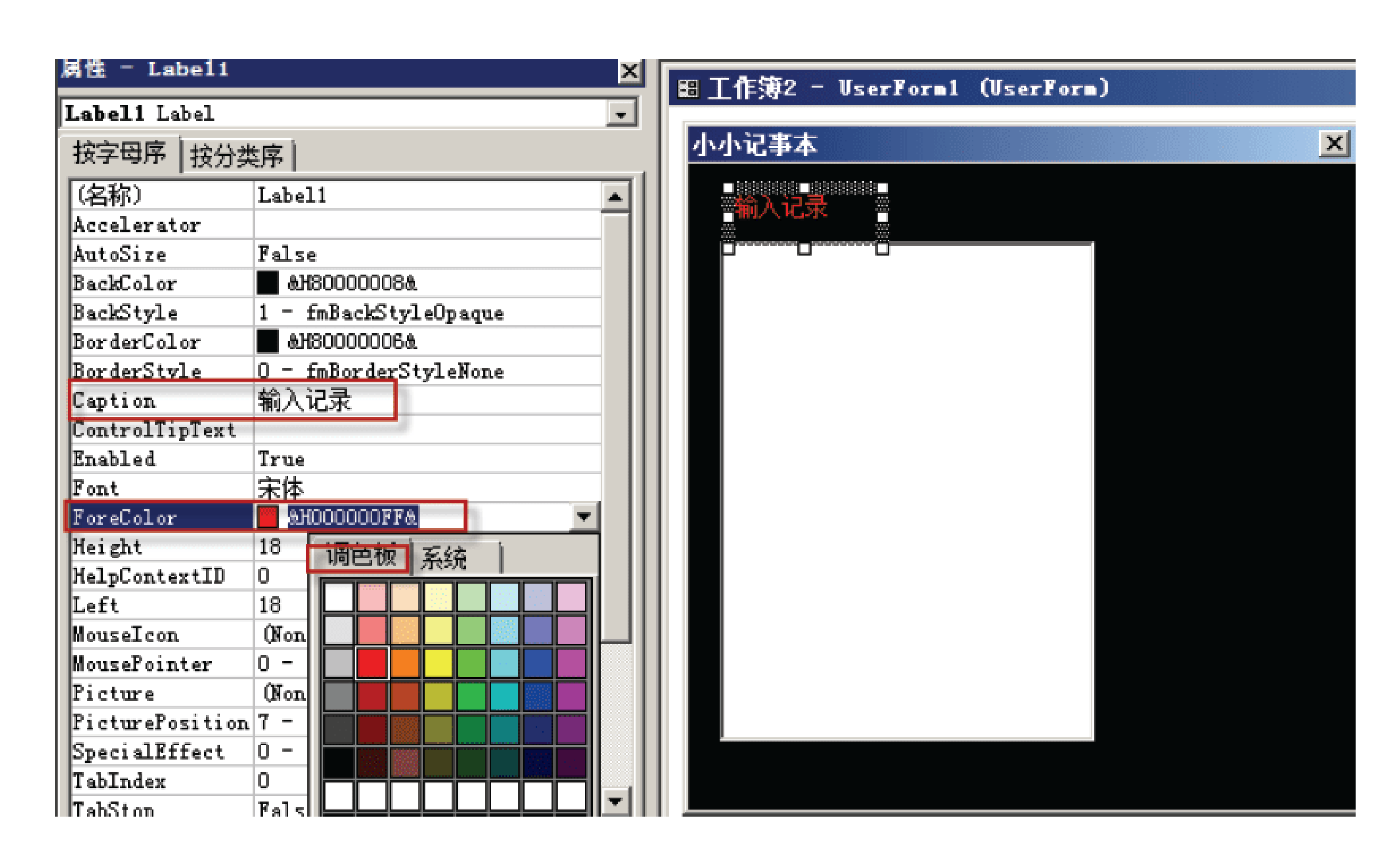


图 15.35　设置“标签”控件的 Caption 属性和 ForeColor 属性

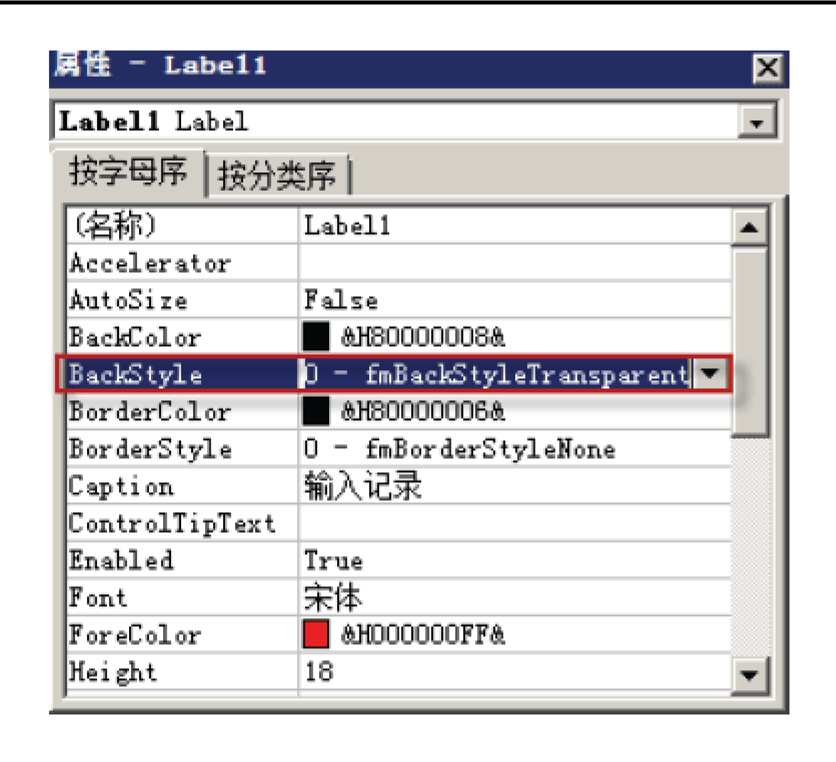


图 15.36　设置“标签”控件属性

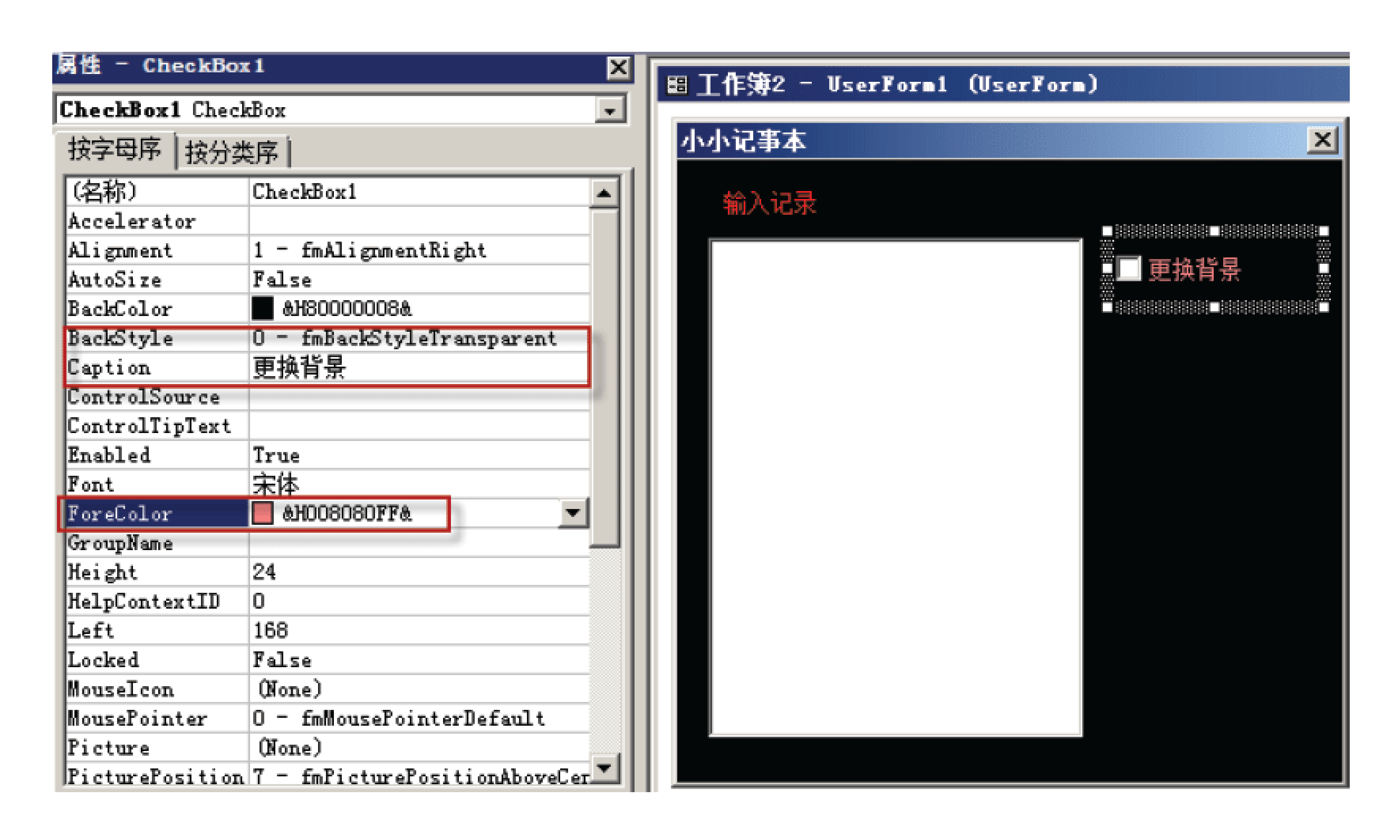


图 15.37　添加控件并设置属性

（4）在用户窗体中添加 3 个“单选按钮”控件，分别设置控件的 fmBackStyleTransparent 属性、Caption 属性和 ForeColor 属性，如图 15.38 所示。

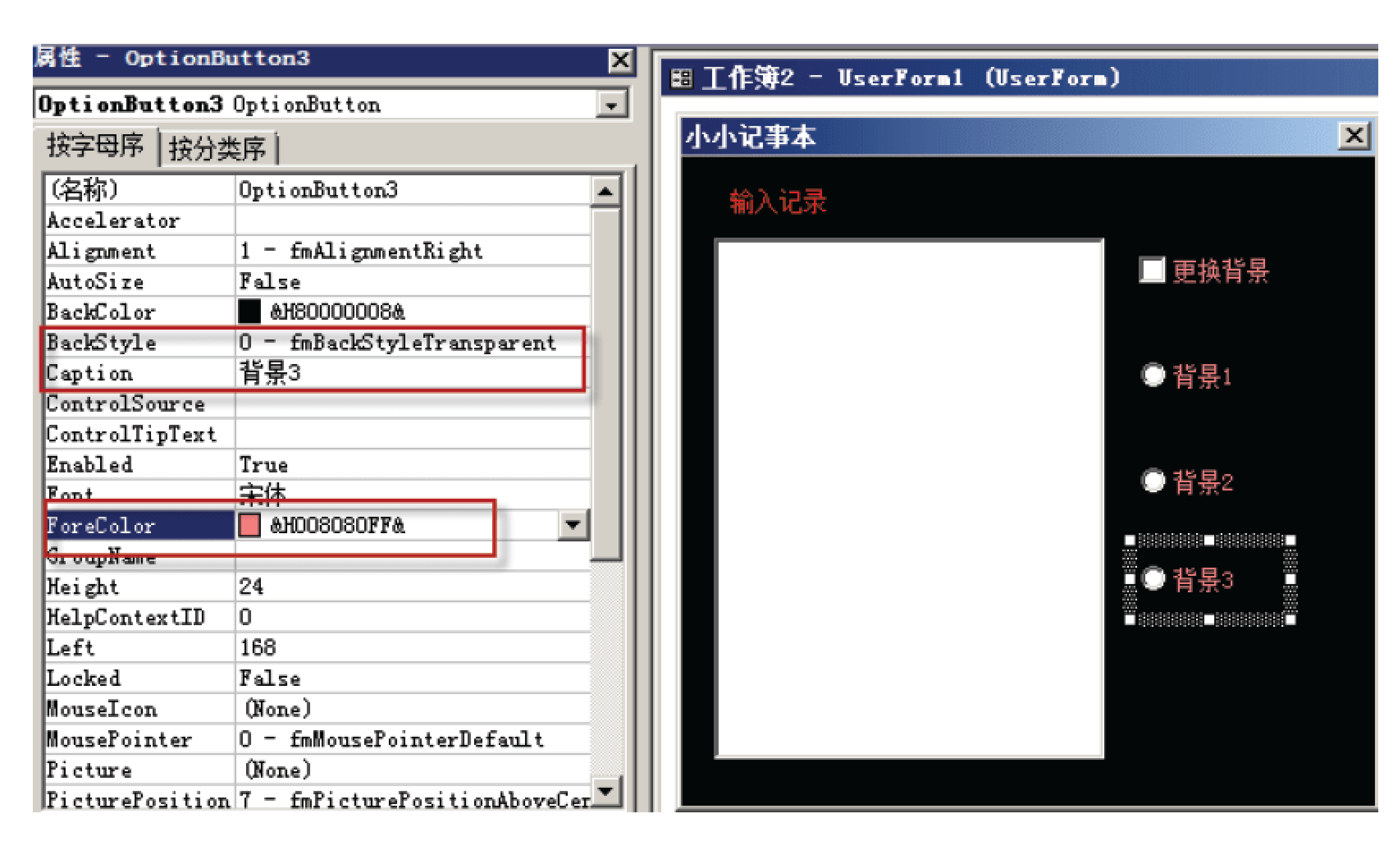


图 15.38　添加“单选按钮”控件并设置属性

（5）双击用户窗体，打开“代码”窗口，为窗体添加 Initialize 事件代码。这里的事件

代码用于窗体中控件的初始化，在打开窗体时，窗体中不使用背景图片，“复选框”控件处于非选择状态，3 个“单选按钮”控件处于不可用状态，如图 15.39 所示。窗体添加 Initialize 事件代码如下所示：

```
Private Sub UserForm_Initialize()
    CheckBox1.Value = False                    '复选框处于非选择状态
    OptionButton1.Enabled = False
    OptionButton2.Enabled = False
    OptionButton3.Enabled = False              '3 个单选按钮均处于非选择状态
    UserForm1.Picture = LoadPicture("")        '不使用背景图片
End Sub
```

（6）在“代码”窗口中为“复选框”控件添加 Click 事件代码。这里，当复选框处于选择状态时，单选按钮才可用，如图 15.40 所示。如果取消复选框的选择，单选按钮将不可用，同时窗体中背景图片取消。代码如下所示：

```
Private Sub CheckBox1_Click()
    If CheckBox1.Value Then                   '如果复选框被选择
        OptionButton1.Enabled = True
        OptionButton2.Enabled = True
        OptionButton3.Enabled = True          '3 个单选按钮可用
        If OptionButton1.Value Then UserForm1.Picture _
        = LoadPicture("背景 1.bmp")  '第一个单选按钮被选择，显示第一张图片
        If OptionButton2.Value Then UserForm1.Picture _
        = LoadPicture("背景 2.bmp")  '第二个单选按钮被选择，显示第二张图片
        If OptionButton3.Value Then UserForm1.Picture _
        = LoadPicture("背景 3.bmp")  '第三个单选按钮被选择，显示第三张图片
    Else                             '复选框取消选择
        UserForm1.Picture = LoadPicture("")      '取消图片加载
        OptionButton1.Enabled = False
        OptionButton2.Enabled = False
        OptionButton3.Enabled = False            '3 个单选按钮不可用
    End If
End Sub
```



图 15.39　控件初始化效果



图 15.40　单选按钮可用

提示：这里，“复选框”控件和“单选按钮”控件的 Value 属性决定了其是否被选择，当其值为 True 时，控件被选择。在程序中使用 If 语句来判断控件的选择情况以

作出不同的反应。在程序中，使用 LoadPicture 方法来指定加载的图片，其参数指定需要载入的图片文件，如果为空则取消图片的加载。

（7）为 3 个“单选按钮”控件添加 Click 事件代码。此时，当选中单选按钮时，可分别加载对应的背景图片，如图 15.41 所示。代码如下所示：

```
Private Sub OptionButton1_Click()
    UserForm1.Picture = LoadPicture("背景1.bmp")    '载入第一张背景图片
End Sub
Private Sub OptionButton2_Click()
    UserForm1.Picture = LoadPicture("背景2.bmp")    '载入第二张背景图片
End Sub
Private Sub OptionButton3_Click()
    UserForm1.Picture = LoadPicture("背景3.bmp")    '载入第三张背景图片
End Sub
```

提示：程序中，当控件发生单击事件时，执行事件程序，将指定图片载入窗体。对于“单选按钮”控件和“复选框”控件来说，Click 事件是常用的事件。



图 15.41　加载背景图片

15.3.4　“列表框”控件

“列表框”控件主要用于显示项目列表，用户可以从中选择一个或多个项目。当项目过多，超过了可显示数目时，列表框会自动添加垂直滚动条，用户可以拖动滚动条查看起初看不见的内容。

在使用“列表框”控件时，需要向控件添加列表项，添加列表项使用控件的 AddItem 方法来实现，其基本语法为：

```
对象名.AddItem[项目字符][,索引号]
```

参数说明：

- 项目字符：该字符用双引号（""）来界定，其值决定该项在列表中的项目名称。
- 索引号：该参数决定项在列表框中的位置。其值是从 0 开始的顺序号，如果没有索引号，则表示新增项目添加到列表的末尾。

如果需要删除控件列表中的某一项，可以使用 RemoveItem 方法，需要删除的项目以索引号来指定。其基本语法为：

```
对象名.RemoveItem[索引号]
```

“列表框”控件除了具有控件的公有属性外，还具有一些其自身独有的属性，其中，List 属性用来获取或设置列表框中的列表项的内容。List 属性是字符串数字，每个数组元素都是列表框中的一个列表项，通过 List（下标）的形式来查询。如，下面语句能够将名为“ListBox1”的列表框控件的第 4 个选项的内容赋予变量 C：

```
C=ListBox1.List(3)
```

在“列表框”控件中，可以使用 ListIndex 属性来判断列表框中哪个列表项被选择。如，选择第一项，ListIndex 的值为 0，选择第二项其值为 1，以此类推。如果没有选择任何一项，则其值为-1。如，当需要删除名为“ListBox1”的“列表框”控件中的被选择项目时，可以使用下面的语句来实现：

```
ListBox1.RemoveItem(ListBox1.Listindex)
```

下面通过一个实例来介绍“列表框”控件的使用方法。该实例模拟一个商品订购系统，用户在能够将左侧“商品条目”列表框中需要的商品选项移到右侧列表框中，同时右侧列表框中不需要的商品选项也可以被重新移回到左侧列表。

（1）打开 Excel 2010 创建一个工作表，在工作表中添加数据，如图 15.42 所示。

（2）新建一个用户窗体，在窗体中添加 2 个“标签”控件、2 个“列表框”控件和 3 个“命令按钮”控件。除了改变控件的 Caption 属性外，其他属性使用默认值。完成后的窗体布局如图 15.43 所示。

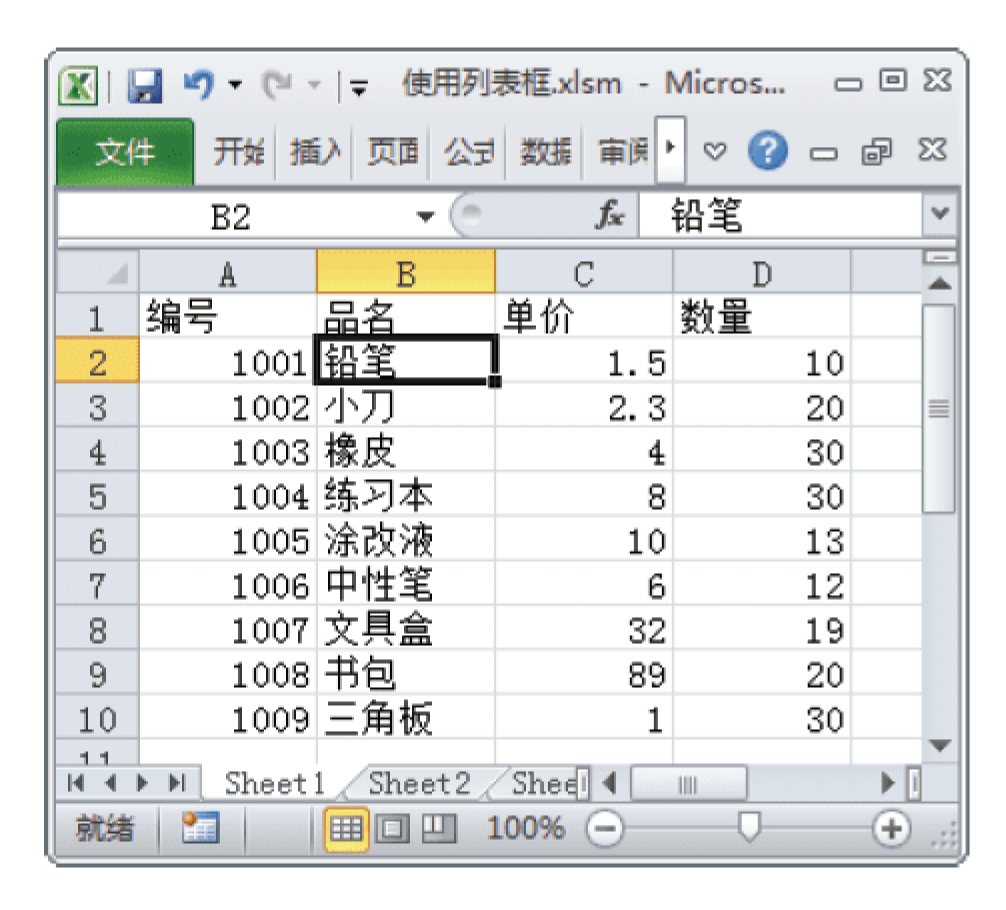


图 15.42　创建工作表

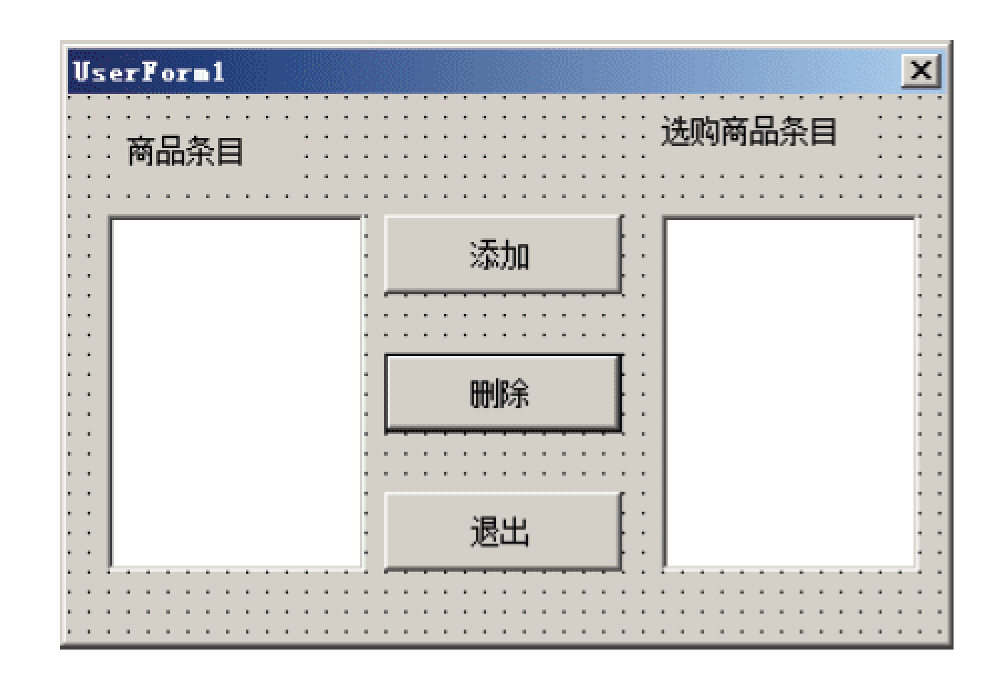


图 15.43　窗体中控件布局

（3）编写窗体 Initalize 事件代码，该事件代码用于控件的初始化。这里，在初始化过程中为第 1 个列表框中添加项目，而将第 2 个列表框中项目清空。窗体初始化后的效果如

图 15.44 所示。窗体的初始化事件代码如下所示：

```
Private Sub UserForm_Initialize()
    Dim i As Long                                          '声明变量
    For i = 2 To Sheet1.Range("B1048576").End(xlUp).Row
                                             '遍历第 2 列中所有有内容的单元格
        ListBox1.AddItem Range("B" & i).Value   '将单元格内容添加到列表中
    Next
        ListBox2.Clear                                  '清空第 2 个列表框
End Sub
```

说明：这里，Sheet1.Range("B1048576").End(xlUp).Row 表示 B 列中所有非空单元格的数目。使用 For…Next 循环结构来遍历工作表中“品名”列中的所有项目，在循环体中使用 AddItem 方法依次将这些项目添加到“列表框”控件中。

（4）为“添加”按钮添加 Click 事件代码。当程序运行时，单击“添加”按钮，在“商品条目”列表中选择的项目将被添加到右侧的列表框中，同时在“商品条目”列表中被删除，如图 15.45 所示。“添加”按钮的事件代码如下所示：

```
Private Sub CommandButton1_Click()
    ListBox2.AddItem ListBox1.List(ListBox1.ListIndex)
                                               '将选择项添加到第二个列表框中
    ListBox1.RemoveItem (ListBox1.ListIndex)  '将选择项从第一个列表中删除
End Sub
```

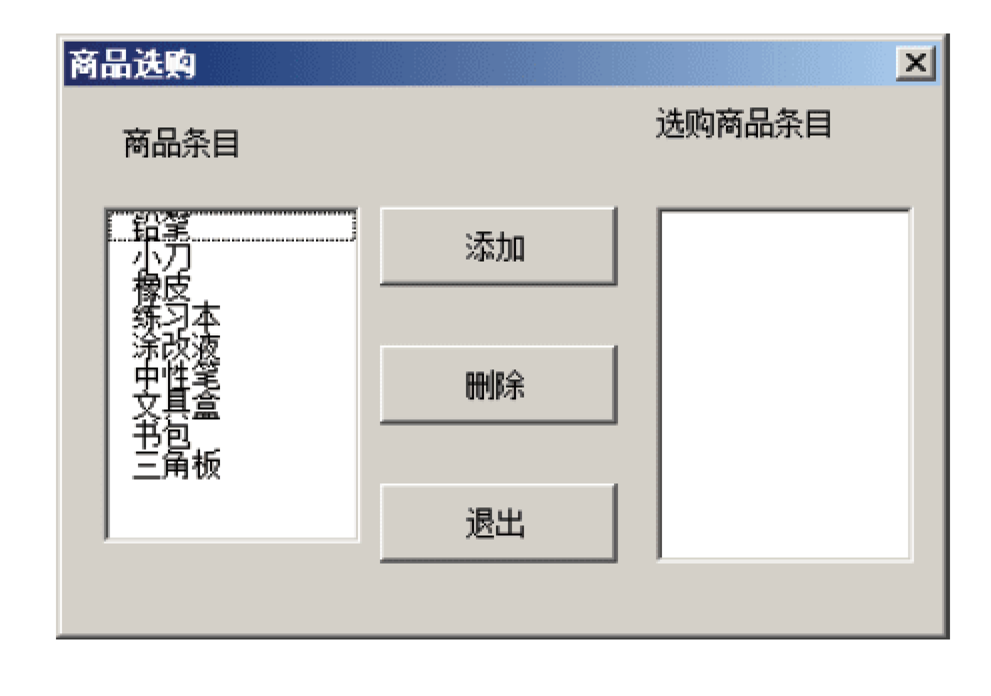


图 15.44　窗体初始化效果

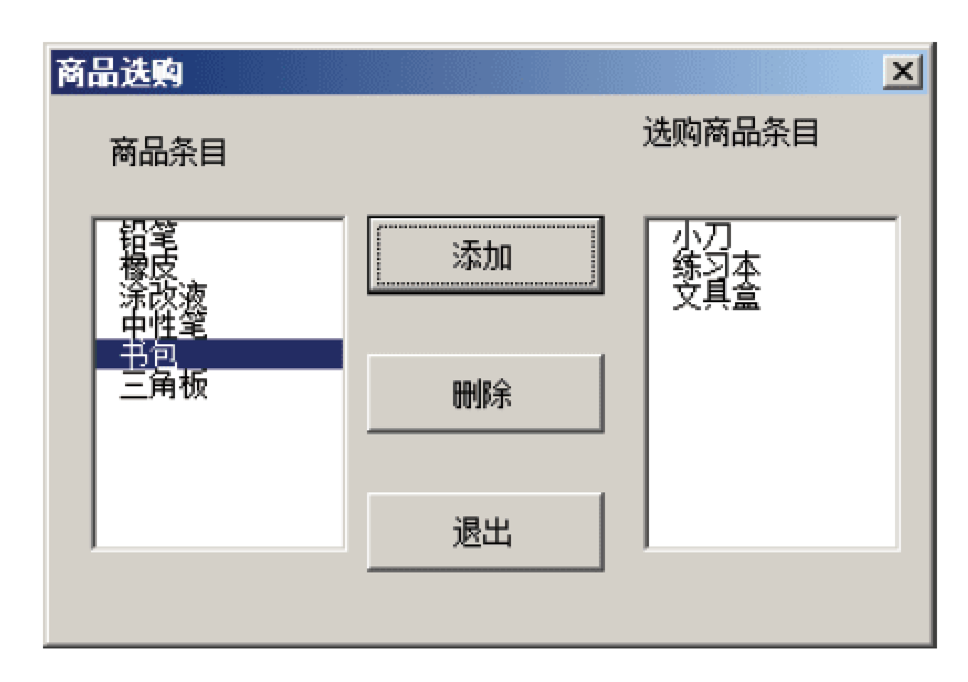


图 15.45　向右侧列表框添加条目

提示：“列表框”控件的 AddItem 方法能够向列表框中添加项目，而 RemoveItem 方法将删除选定的项目。添加或删除的项目是以项目索引号指定，整个索引号可以使用 ListBox1.ListIndex 语句来获得。

（5）为“删除”按钮添加 Click 事件代码。程序运行时，单击“删除”按钮能够将右侧“选购商品条目”中的选项删除。如，删除右侧列表中的“小刀”选项，如图 15.46 所示。“删除”按钮的事件代码如下所示：

```
Private Sub CommandButton2_Click()
    ListBox1.AddItem ListBox2.List(ListBox2.ListIndex)
                                               '将选择项添加到第一个列表框中
    ListBox2.RemoveItem (ListBox2.ListIndex)  '将选择项从第二个列表中删除
End Sub
```

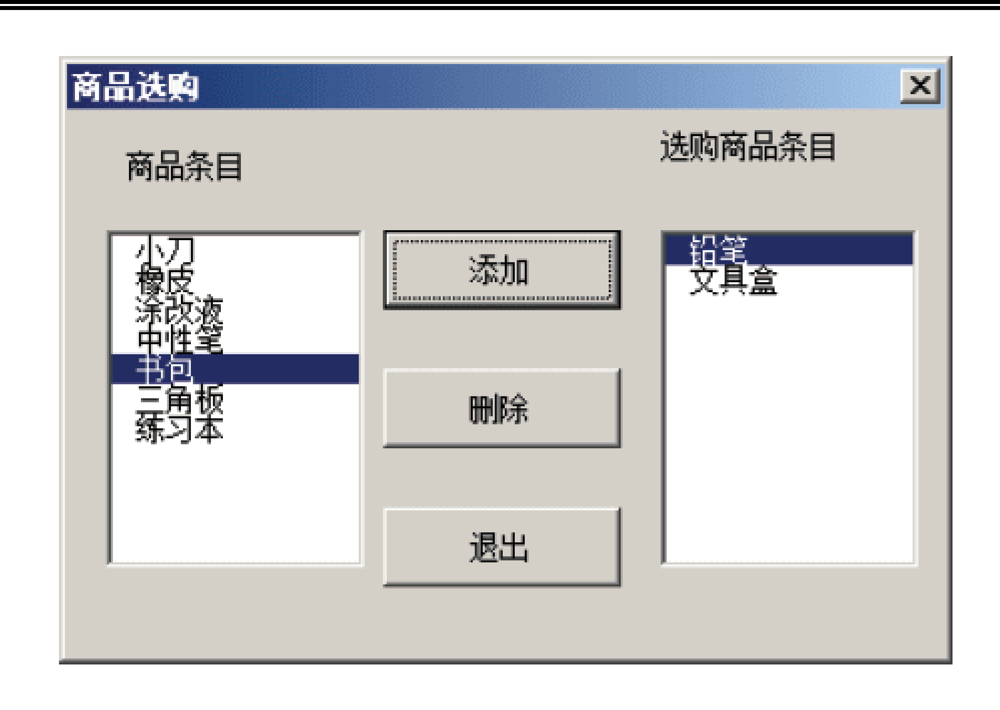


图 15.46　删除“小刀”选项

（6）为“退出”按钮添加 Click 事件代码。程序运行时，单击“退出”按钮，将退出窗体。“退出”按钮的事件代码如下所示：

```
Private Sub CommandButton3_Click()
    Unload Me                                               '卸载窗体
End Sub
```

15.3.5　“组合框”控件

“组合框”控件用于从多个项目中选择一个项目，其具有与“列表框”控件基本相同的控件属性和方法。下面以一个实例来介绍“组合框”控件的使用方法。该实例能够通过窗体中的“组合框”控件来获得工作表中的商品名，用户选择后，选择商品的商品信息将在其下的“列表框”控件中显示。

（1）打开 15.3.4 节使用的工作表。重新创建用户窗体，在用户窗体中添加 1 个“标签”控件、1 个“列表框”控件、2 个“命令按钮”控件和 1 个“组合框”控件。设置控件的名称和它们在窗体中的大小和位置，如图 15.47 所示。

（2）为窗体添加 Initialize 事件代码。这里的事件代码向“组合框”控件中添加项目，同时对“列表框”控件进行初始化。“组合框”控件添加项目后的效果如图 15.48 所示。Initialize 事件代码如下所示：

```
Private Sub UserForm_Initialize()
    Dim i As Long                       '声明变量
    For i = 2 To Sheet1.Range("B1048576").End(xlUp).Row
                                        '遍历第 2 列中所有有内容的单元格
        ComboBox1.AddItem Range("B" & i).Value
                                        '将单元格内容添加到组合框中
    Next
        ListBox1.Clear                  '清空列表框
End Sub
```

提示： 这里，使用 For…Next 循环来实现对工作表中指定单元格的遍历，在循环体中使用“组合框”控件的 AddItem 方法来将单元格的内容添加到控件的列表中。

（3）为“组合框”控件的 Change 事件添加代码。这里，组合框中选择的选项将添加到列表框中，如图 15.49 所示。“组合框”控件的 Change 事件代码如下所示：

```
Private Sub ComboBox1_Change()
    ListBox1.AddItem ComboBox1.Value    '将组合框中的选项添加到列表框中
End Sub
```

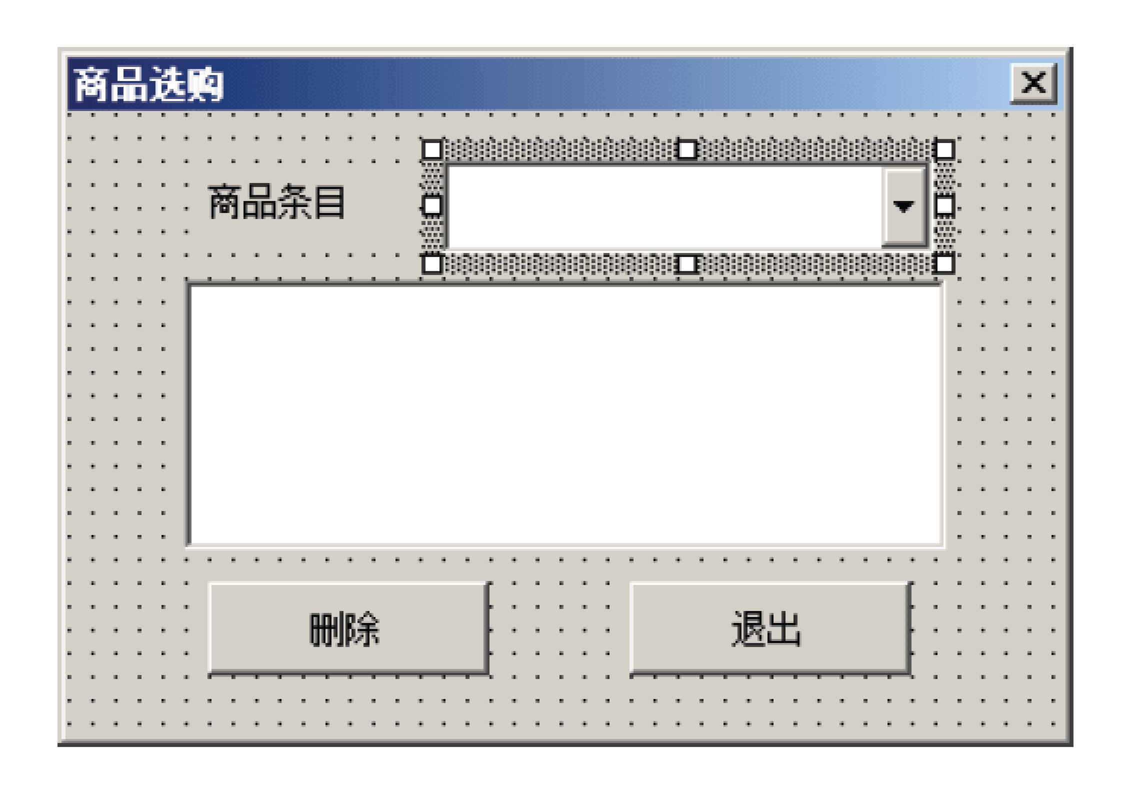


图 15.47　添加控件



图 15.48　向组合框中添加项目

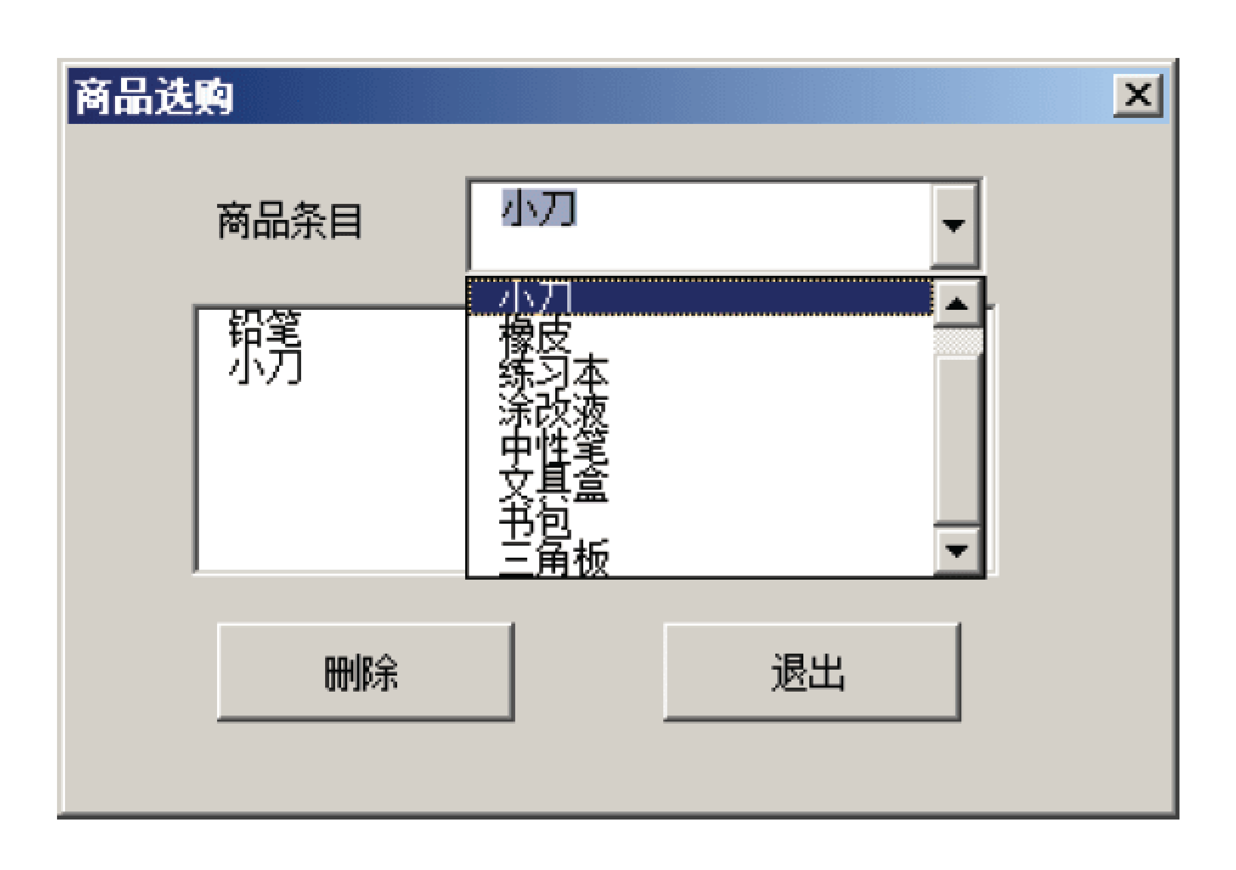


图 15.49　将组合框中的选项添加到列表框中

（4）为“删除”按钮和“退出”按钮添加 Click 事件代码。这两个按钮的功能是删除列表框中的选项和卸载窗体。两个控件的事件代码如下所示：

```
Private Sub CommandButton1_Click()
    ListBox1.RemoveItem (ListBox1.ListIndex)  '将选择项从第一个列表中删除
End Sub
Private Sub CommandButton2_Click()
    Unload Me                                 '卸载窗体
End Sub
```

15.3.6　“图像”控件和“数字调节钮”控件

“图像”控件实际上是一个图片的容器，可以使图片作为数据的一部分在窗体中显示出来。在使用“图像”控件时，控件能够对图片进行裁剪、调整大小以及缩放，但无法对图片进行编辑，如，无法改变图像的色彩、色调或为图片添加特效等。“图像”控件可以支持常见的图像格式，如*bmp、*.cur、*.gif、*.ico、*.jpg 和*.wmf 等。

在“图像”控件中，Picture 属性指定需要显示的图片文件名。在设计窗体时，该属性

可以在“属性”对话框中进行设置以指定需要显示的图片。在运行窗体时，该属性使用 LoadPicture 函数来指定，其语法格式如下：

```
对象名.Picture=LoadPicture(pathName)
```

其中，对象名是“图像”控件的控件名，pathName 参数是一个图片文件的完整路径。

载入“图像”控件的图片，可以使用 PictureAlignment 属性来设置其在控件中的对齐方式，如果需要图像左上角与控件左上角对齐，可将其值设置为 fmPictureAlignment TopLeft。如果需要图片的中心与控件中心对齐，可将其设置为 fmPictureAlignmentCenter。当其值为 fmPictureAlignmentBottomLeft 时，图片左下角与控件左下角对齐。

载入控件的图片的大小可以通过设置 PictureSizeMode 属性值来调整。如，将其设置为 fmPictureSizeModeClip 时，图片中比控件大的部分被裁剪。而如果希望图片将拉伸以填满整个控件，可将其值设置为 fmPictureSizeModeStretch。

“数字调节钮”控件（即“旋转按钮”控件）主要用来输入移动范围内的整数值。单击控件的调节按钮能够更改数值，使用该控件能够对控件或对象的某些数值进行调整。这个控件在前面多次使用，这里就不详细介绍了。

下面通过一个简单的图像浏览器的制作，来介绍“图像”控件和这个“数字调节钮”控件的使用方法。在实例中，窗体中显示图片，单击“数字调节钮”控件实现图片显示翻页，并且图片能够循环显示。

（1）准备 5 张用来演示的素材图片，其文件名为“1.bmp”～“5.bmp”。新建一个用户窗体，在窗体中添加 1 个“图像”控件、1 个“数字调节按钮”控件和 2 个“标签”控件。窗体和“标签”控件设置标签文字，其他控件的属性使用默认值即可，控件在窗体中的布局如图 15.50 所示。

（2）编写窗体的 Initialize 事件代码。在 Initialize 事件代码中，需要设置“数值调节钮”控件的最大值和最小值以及其初始值。同时，对“图像”控件初始化，使“图像”控件显示第一张图片，如图 15.51 所示。

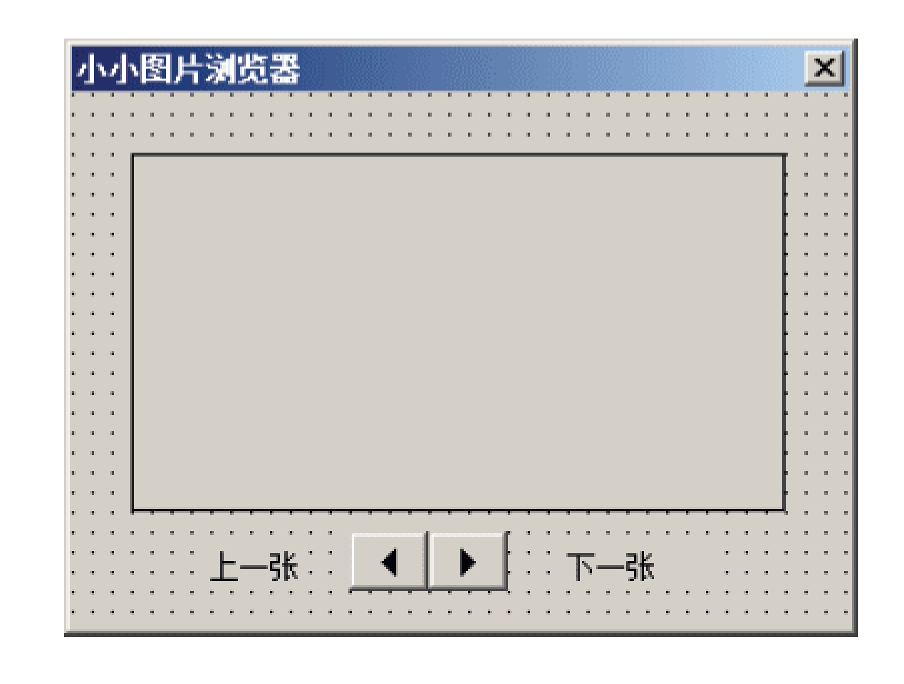


图 15.50　窗体中的控件布局



图 15.51　显示第一张图片

窗体的事件代码如下所示。

```
Private Sub UserForm_Initialize()
    SpinButton1.Max = 6                                        '设置控件最大值
    SpinButton1.Min = 0                                        '设置控件最小值
    SpinButton1.Value = 1                                      '设置控件当前值
    Image1.PictureSizeMode = fmPictureSizeModeStretch          '图片拉伸显示
    Image1.Picture = LoadPicture(SpinButton1.Value & ".bmp")
```

```
                                                       '显示第一张图片
07   End Sub
```

提示：这里，Min 属性用于指定“数值调节钮”控件的 Value 属性的最小值，Max 属性指定控件 Value 的最大值。在程序运行时，Value 属性值将随着单击控件按钮而发生改变。在第 06 行代码中，使用 LoadPicture 方法来实现图片的载入，其中载入文件的文件名是编号数字，“数值调节钮”控件的 Value 值作为文件名来指定需要载入的图片。

（3）下面为“数值调节钮”控件添加 Change 事件代码。该事件是在单击控件上的按钮引起 Value 值改变时发生。在程序运行时，通过单击“数值调节钮”控件上的按钮改变“图像”控件中显示的图像，就像书的翻页那样。同时图片的浏览可以循环进行。这里，“数值调节钮”控件的 Change 事件代码如下所示。

```
01   Private Sub SpinButton1_Change()
02       If SpinButton1.Value <= 5 And SpinButton1.Value >= 1 Then
                                                       '控件值在 1～5 之间
03           Image1.Picture = LoadPicture(SpinButton1.Value & ".bmp")
                                                       '显示对应图片
04       ElseIf SpinButton1.Value = 6 Then             '控件值为 6
05           SpinButton1.Value = 1                     '从第一张图片开始
06       Else
07           SpinButton1.Value = 5                     '从第五张图片开始
08       End If
09   End Sub
```

提示：这里，当“数值调节钮”控件的值在 1～5 时，载入指定的图片。如果控件的值超过了 5，说明 5 张照片已经依次显示完，此时应该再重新显示第一张照片，因此程序的第 05 行将控件的 Value 值设置为 1。同理，当其值为最小值 0 时，则返回重新显示编号最大的图片，因此在第 07 行将 Value 值设置为 5。

15.3.7　TabStrip 控件和“多页”控件

TabStrip 控件允许用户使用相同的控件来显示多套相同格式的数据。TabStrip 控件是由两个部分构成，它们是标签和客户区，标签用于选择打开的客户区，客户区用于放置各种控件。TabStrip 控件实际上可以理解为包含一个或多个标签的集合，每一个标签都是一个可供选择的单独的对象，而客户区由所有的标签共享。

“多页”控件和 TabStrip 控件相似，“多页”控件可以在窗体中显示一系列不同的页面，可用来处理需要划分不同类别的信息。如，使用“多页”控件来显示职工信息时，可以第一页显示职工基本信息，如姓名、住址和身份证号等，第二页显示职工的学历、毕业学校和工作年限等，在第三页显示职工的工作经历。

“多页”控件对象为 MultiPage，其包含 Page 对象（即页对象）。每个页对象类似于一个单独的窗体，都可以包含一套自己的控件，并且不需要依赖集合中的其他页对象的数据。在默认的情况下，每个“多页”控件包含两个页面，每个页面就是一个 Page 对象，它们构

成了多页的 Pages 集合。

TabStrip 控件和“多页”控件在使用方法、属性和功能上有很多相同的地方，如控件的 MultiRow 属性设置是否显示多行标签，Value 属性表示当前激活的页，ScrollBars 属性可设置页面是否有垂直或水平滚动条等。

作为两个不同的控件，它们最大的区别在于作用上的不同。TabStrip 控件每个页允许使用相同的控件来实现对多套不同格式数据的处理，其客户区不是独立的。“多页”控件每个页面都是独立的客户区，每个页中使用独立的控件来对数据进行处理。显然，当使用的是单一数据布局，可以选择 TabStrip 控件。而如果是多数据布局，如，像 Excel 2010 的功能区那样，在不同的页面中使用不同的控件，则必须使用“多页”控件。

下面通过一个实例来介绍 TabStrip 控件的使用方法和对该控件编程的技巧。本实例用于在窗体中显示书籍的篇章目录结构。当程序运行时，单击窗体中的标签，在选项卡中能够显示对应章节目录。



图 15.52　在窗体中添加一个 TabStrip 控件

（1）创建一个窗体，在窗体中添加一个 TabStrip 控件，如图 15.52 所示。将窗体的 Caption 属性设置为“《Excel 2010VBA 从入门到精通》篇章结构”，TabStrip 控件的属性使用默认值即可。

（2）在控件的“Tab1”标签上右击，在弹出的快捷菜单中选择 “重命名”命令，打开“重命名”对话框。在对话框的“题注”文本框中输入文字，然后单击“确定”按钮，如图 15.53 所示。关闭“重命名”对话框后，标签名将会更改。采用相同的方法更改“Tab2”标签名，效果如图 15.54 所示。

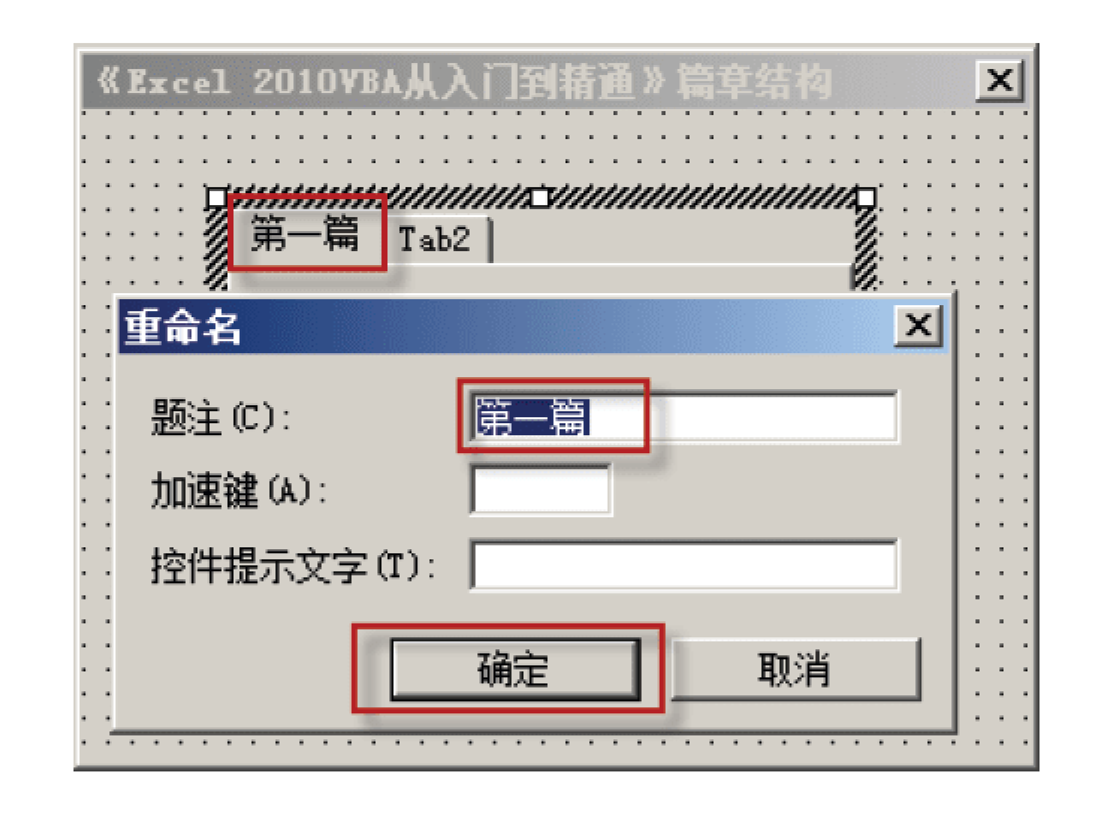


图 15.53　“题注”文本框中输入标签名

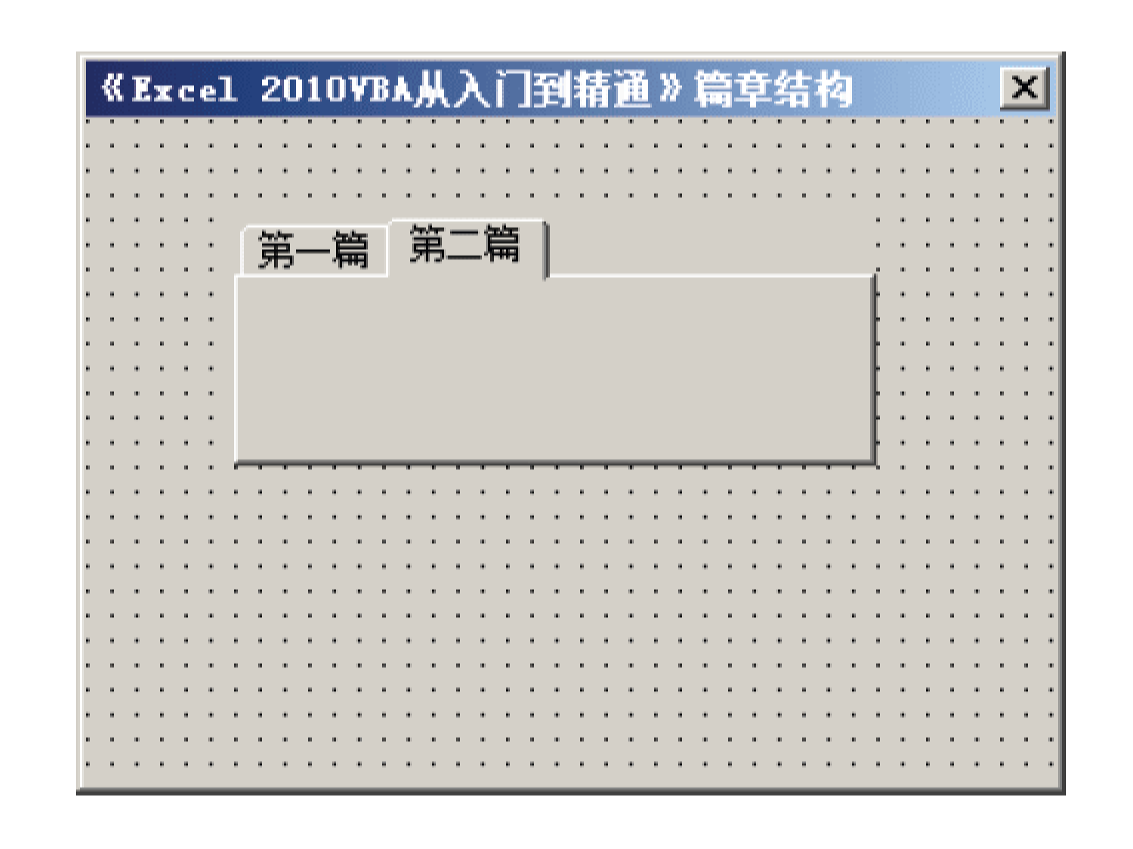


图 15.54　更改标签名后的效果

（3）在标签上右击，然后在弹出的快捷菜单中选择“新建页”命令，如图 15.55 所示。此时，可以在选择标签的后方添加一个新标签。再添加 3 个标签，将它们分别命名为“第三篇”、“第四篇”和“第五篇”，如图 15.56 所示。

提示：在标签的右键快捷菜单中，选择“删除页”命令将删除当前标签。选择“移动”命令可以打开“页面顺序”对话框，使用该对话框可以调整页面的排列顺序。

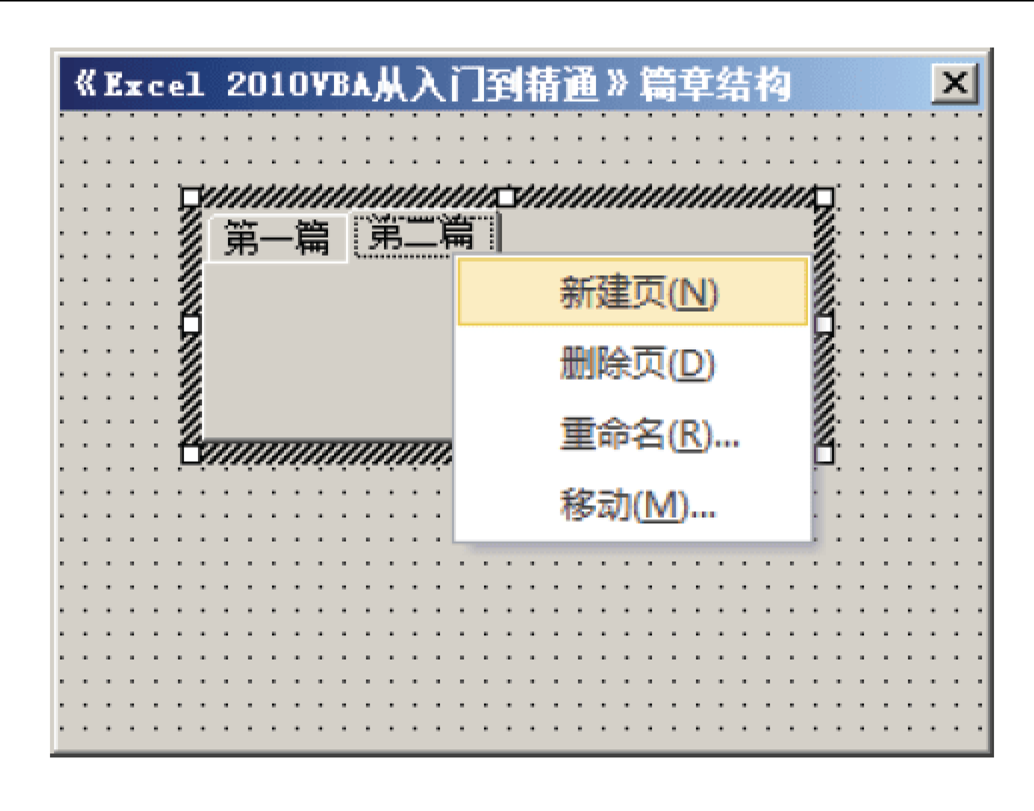



图 15.55　选择“新建页”命令

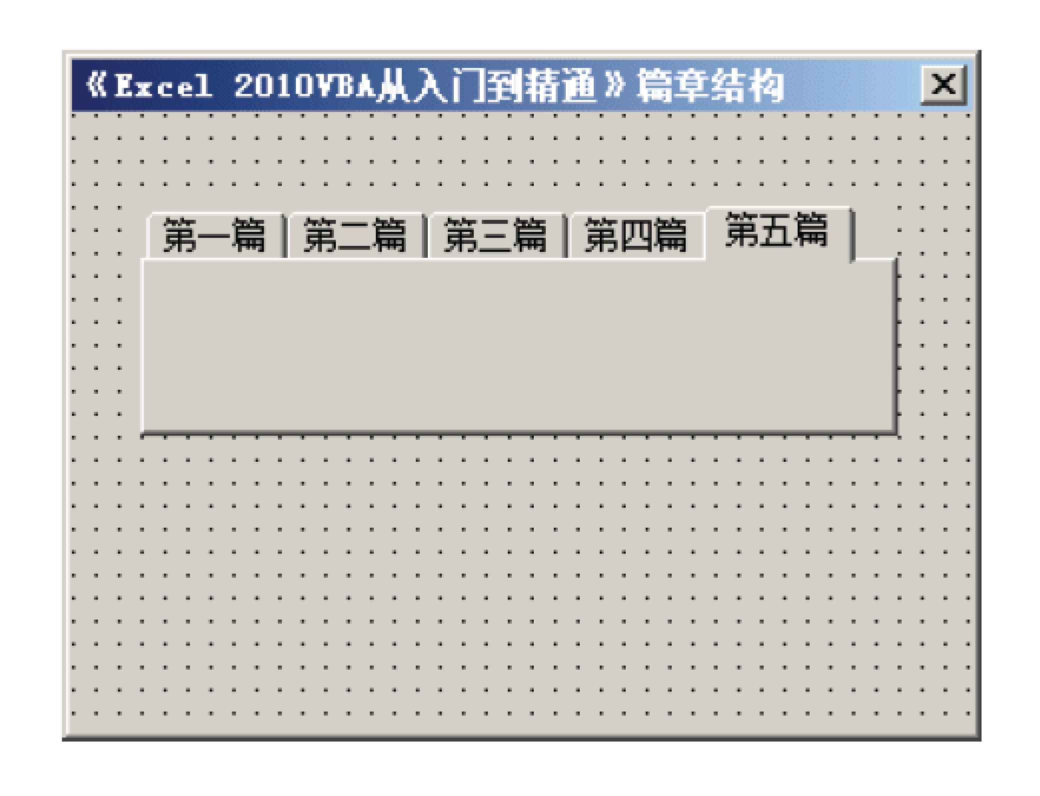


图 15.56　添加 3 个标签

（4）在控件中添加一个“文本框”控件，如图 15.57 所示。在“属性”对话框中，将控件的 MultiLine 属性设置为 True，如图 15.58 所示。

警告：这里，必须将 MultiLine 属性设置为 True，否则在程序中即使使用了换行符，程序运行时文本框中的文字也不会换行。

（5）在 TabStrip 控件添加 Change 事件代码，该事件当控件中标签的选择发生改变时激发。这里，要求选择不同的标签时，客户区中文本框显示相应的内容，如图 15.59 所示。



图 15.57　添加“文本框”控件

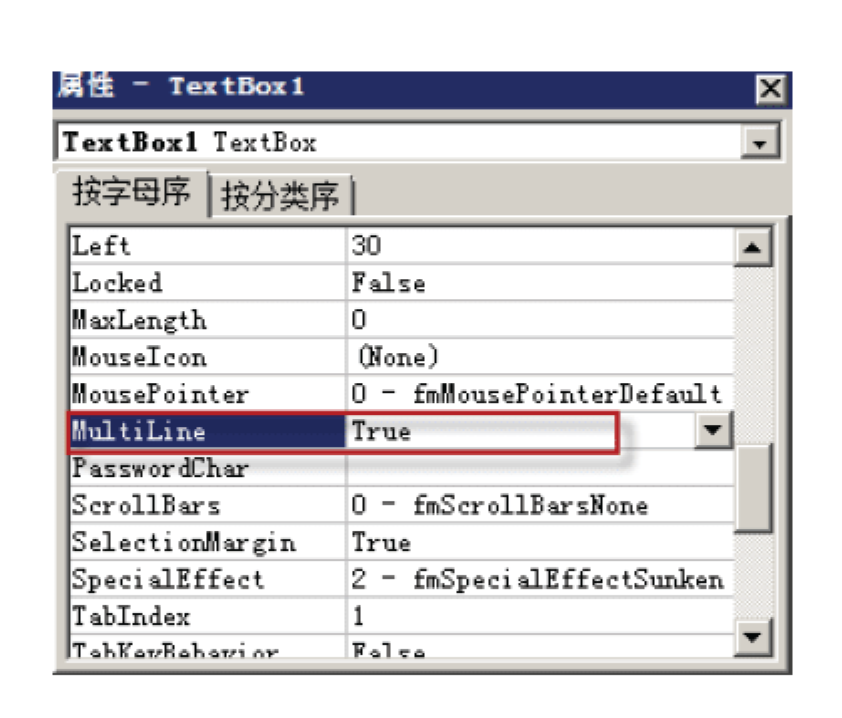


图 15.58　设置控件的 MultiLine 属性

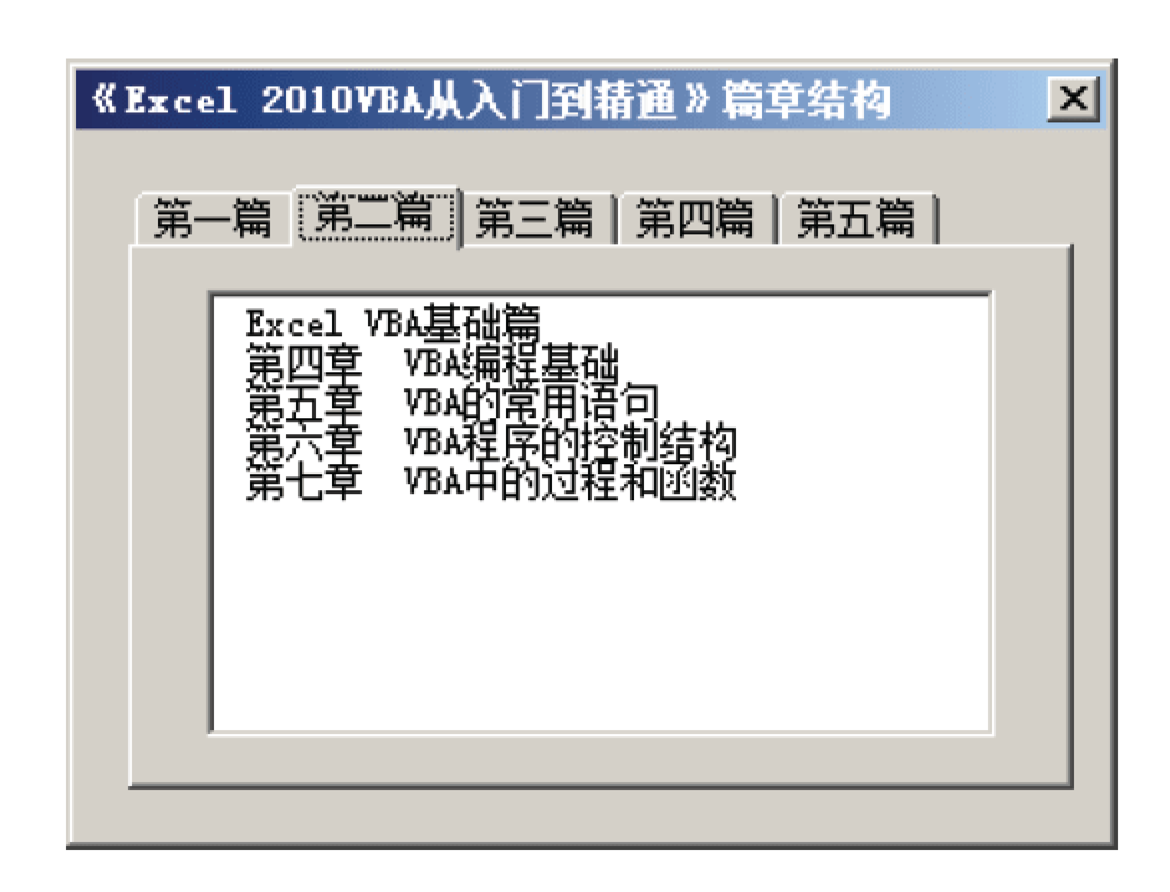


图 15.59　选项卡的文本框中显示相应的内容

在控件的一个标签上双击打开“代码”窗口，输入如下程序代码：

```
Private Sub TabStrip1_Change()
    Select Case TabStrip1.SelectedItem.Index            '确定选择的标签
    Case 0                                              '选择第一个标签
        TextBox1.Text = "Excel VBA 准备篇" & Chr(13) _
        & "第一章  开始 Excel VBA 编程之旅--认识宏" & Chr(13) _
        & "第二章  VBA 概述" & Chr(13) _
        & "第三章  第一个 VBA 程序"                     '文本框中显示对应的内容
    Case 1                                              '选择第二个标签
        TextBox1.Text = "Excel VBA 基础篇" & Chr(13) _
        & "第四章  VBA 编程基础" & Chr(13) _
        & "第五章  VBA 的常用语句" & Chr(13) _
        & "第六章  VBA 程序的控制结构" & Chr(13) _
        & "第七章  VBA 中的过程和函数" '文本框中显示对应的内容
    Case 2                                              '选择第三个标签
        TextBox1.Text = "Excel VBA 对象模型篇" & Chr(13) _
        & "第八章  Excel 的对象模型" & Chr(13)
        & "第九章  VBA 常用对象应用之一--Application 对象使用" & Chr(13)
        & "第十章  VBA 常用对象应用之二--使用工作簿对象" & Chr(13)
        & "第十一章 VBA 常用对象应用三--操作工作表" & Chr(13)
        & "第十二章    VBA 常用对象应用三--操作单元格"
                                                        '文本框中显示对应的内容
    Case 3                                              '选择第四个标签
        TextBox1.Text = "Excel VBA 用户界面设计篇" & Chr(13)
        & "第十三章   创建工作表操作界面" & Chr(13)
        & "第十四章  创建用户窗体" & Chr(13)
        & "第十五章    自定义 Excel 2010 功能区"        '文本框中显示对应的内容
    Case 4                                              '选择第五个标签
        TextBox1.Text = "Excel VBA 专项操作篇" & Chr(13) _
        & "第十六章    使用 VBA 控制图表" & Chr(13) _
        & "第十七  使用类模块来创建对象" & Chr(13) _
        & "第十八章    VBA 的数据库编程"                '文本框中显示对应的内容
    End Select
End Sub
```

提示： TabStrip 控件使用时，可以使用 SelectItem.Index 语句获得控件选择标签的索引号。程序中使用 Select Case 结构来对标签的选择进行判断，根据具体的选择在“文本框”控件中显示对应的内容。本例如果是使用“多页”控件，则需要分别向每个页中添加 1 个文本框控件才能实现各篇目录的显示。有兴趣的读者可以自己尝试一下使用“多页”控件来实现上面实例的功能。

15.3.8　Refedit 控件

在使用 Excel 函数时，少不了要引用单元格，比较简单的做法是使用“参照”按钮来实现对单元格的选择。如，在使用 Max 函数确定单元格区域中数据的最大值时，可以采用下面的操作：

（1）打开“函数参数”对话框，如图 15.60 所示。

（2）单击对话框中的按钮将对话框折叠，然后可以通过在工作表中拖动光标来选择需要求最大值的单元格区域，如图 15.61 所示。

（3）再次单击按钮，返回“函数参数”对话框，关闭对话框即可使用函数获得需要的结果，这样的功能为单元格的选取提供了方法。在 VBA 应用程序中，使用 Refedit 控件同样也能实现这种功能。

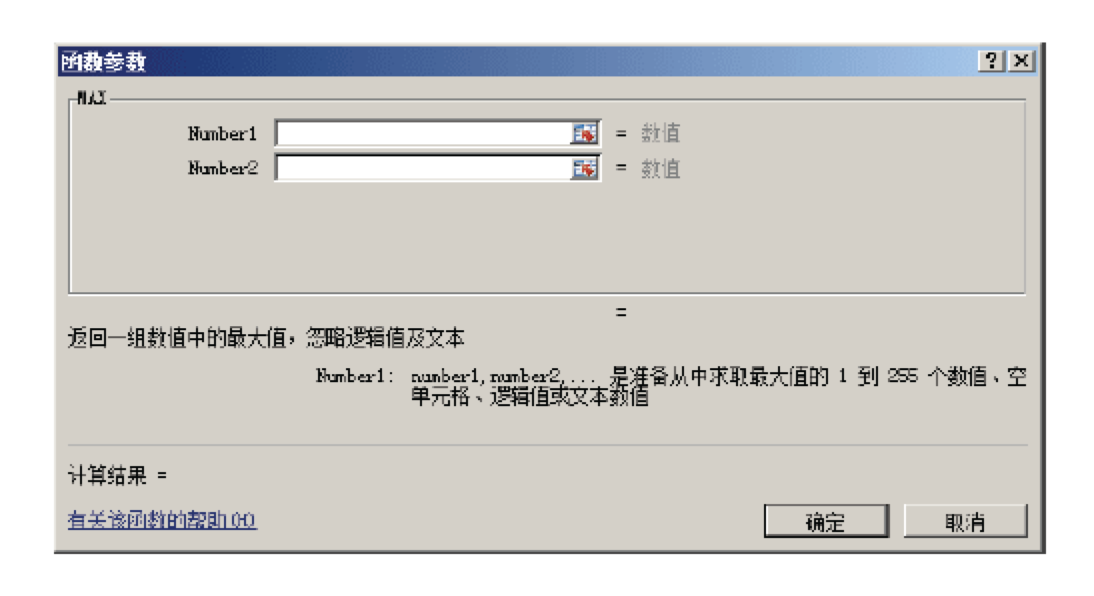


图 15.60　“函数参数”对话框

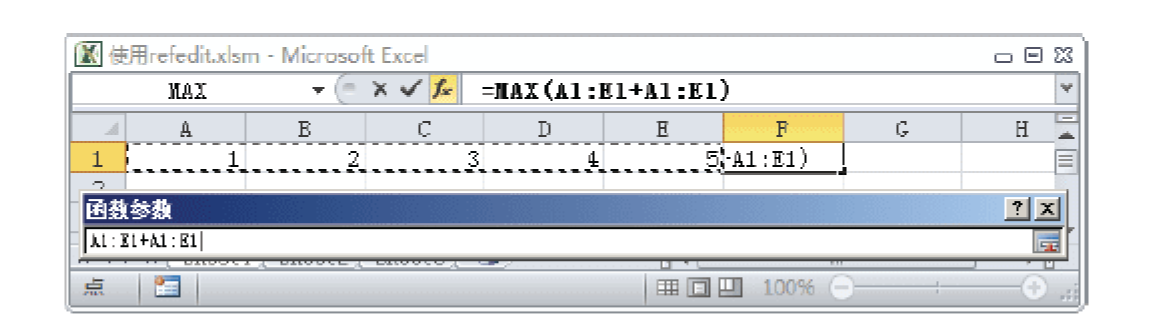


图 15.61　选择单元格

下面以一个实例来介绍 Refedit 控件的使用方法。该实例使用 Refedit 控件来选择工作表中单元格，以实现对选择单元格中数据的格式的转换。

（1）打开需要处理的 Excel 工作表。在 Visual Basic 编辑器中新建一个用户窗体，在用户窗体中添加 2 个“标签”控件、1 个“命令按钮”控件和 1 个 Refedit 控件，如图 15.62 所示。对“标签”控件、“命令按钮”控件和用户窗体的属性进行修改，同时调整控件的大小和位置，完成后窗体中控件的布局如图 15.63 所示。

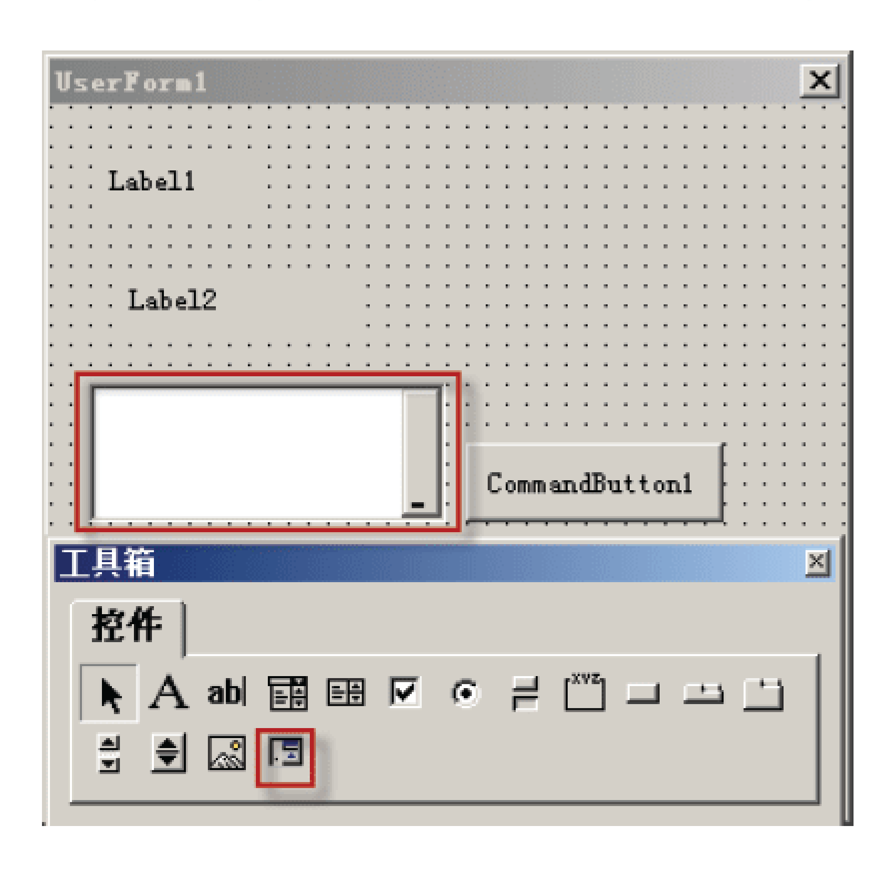


图 15.62　添加控件

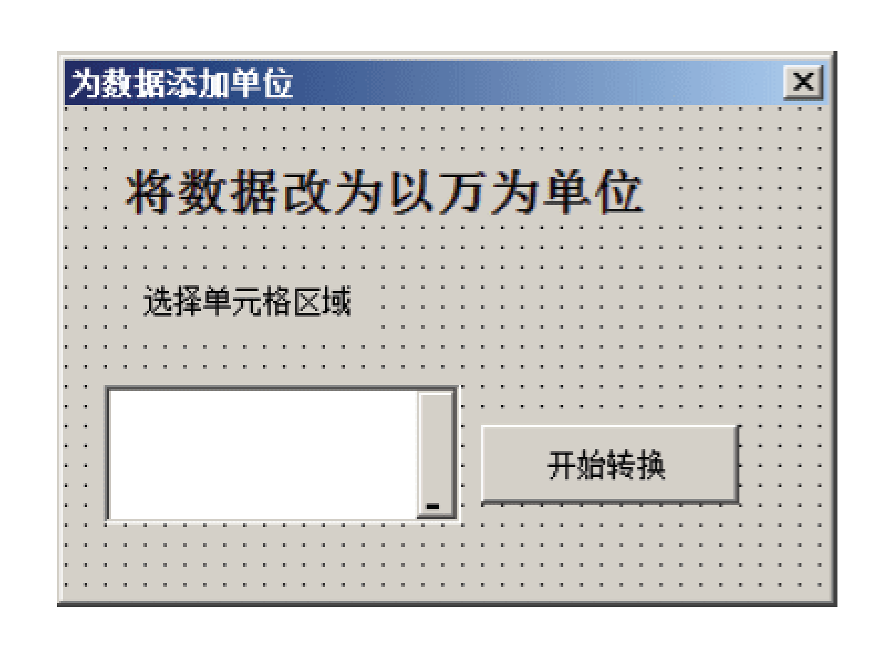


图 15.63　窗体中控件的布局

（2）双击窗体中“命令按钮”控件，打开“代码”窗口为控件添加 Click 事件代码。该事件代码将使用 Refedit 控件获取的单元格中数据转换为以“万”为单位的形式。“命令按钮”控件的事件代码如下所示：

```
Private Sub CommandButton1_Click()
    Dim rng As Range                     '声明一个对象变量
    Set rng = Range(RefEdit1.Value)      '将选择单元格对象赋予变量
    rng.NumberFormatLocal = "#"".""#,万"
                                         '将选择单元格中数据改为以万为单位
End Sub
```

提示：在程序中，RefEdit1.Value 语句通过控件的 Value 属性获得用户选择的单元格的地址。NumberFormatLocal 是 Range 对象属性，用于设置单元格数据格式。这里的格式代码，可以参看 Excel 2010 的“设置单元格格式”对话框中“自定义”选项“类型”列表中的格式样式。

（3）按 F5 键启动程序，打开的用户窗体如图 15.64 所示。单击 Refedit 控件上的按钮，用户窗体收缩显示，在单元格中选择需要的单元格，如图 15.65 所示。



图 15.64　打开的用户窗体

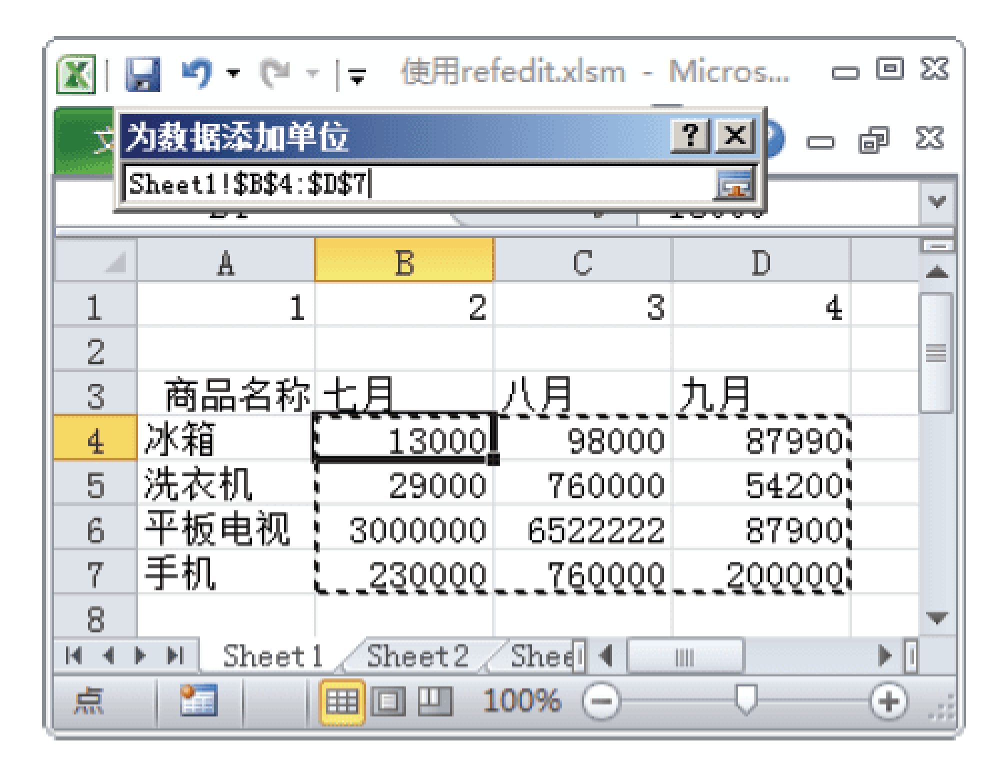


图 15.65　选择单元格

（4）完成单元格选择后，单击按钮回到用户窗体。此时 Refedit 控件中显示选中单元格的地址，单击窗体中的“开始转换”按钮，被选中的单元格中数据格式转换为需要设定的格式，如图 15.66 所示。

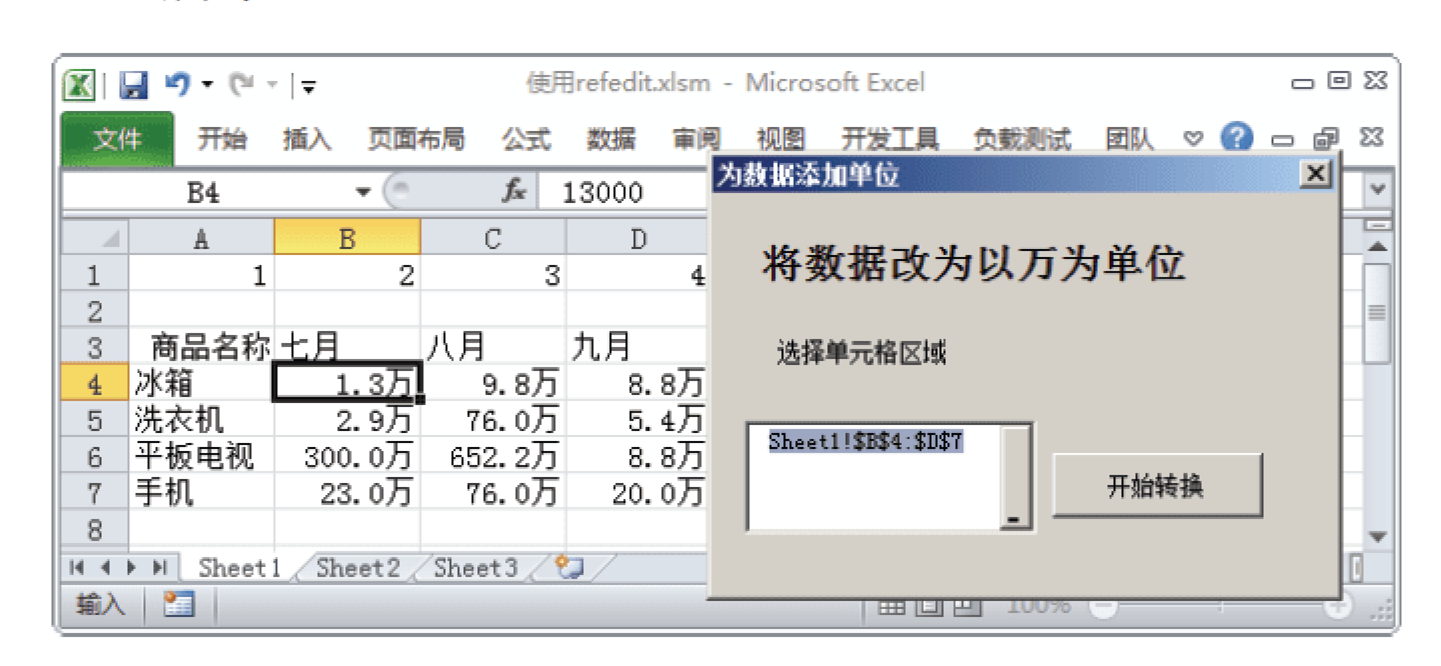


图 15.66　选中单元格区域中的数据格式被改变

15.4　使用 Excel 附加的 ActiveX 控件

附加控件指的是那些并没有出现在控件工具箱中、使用前需要将其载入控件工具箱的控件。附加控件的添加方法在本章的 15.2.1 节中已经作了详细介绍。本节将重点介绍几个典型的附加控件的使用方法。

15.4.1　ListView 控件

ListView（即列表视图）控件是用于显示项目列表的控件，其能够使用 4 种不同的视

图来显示项目。在项目列表中，项目被组成带有或不带有列标头的列，项目除了是传统的文本外，还可以带有图标。

ListView 控件包括 3 种视图方式，可以通过设置 Arrange 属性值来设置。当其值为 0 时，表示不排列；当其值为 1 时，表示左对齐，项目自动沿着控件的左侧对齐；当其值为 2 时，表示顶端对齐，项目自动沿控件的顶端对齐。

ListView 控件具有两种报表视图属性，使用 View 属性来进行设置。当该属性值为 lwReport 时，是报表型；其值为 lwList 时，是列表型。

在使用 ListView 控件时，有时需要更改列表项的排列顺序。控件的 Sorted 属性决定了项目是否可以排序，其值为 True 允许排序。控件的 SortKey 属性决定了对象排序的关键字。控件的 SortOrder 属性决定对象排序的顺序，当其值为 0 时，按升序排列。当其值为 1 时，按降序排列。

使用 ListView 控件时常常会对选项进行选择，控件能够对鼠标的单击事件做出响应。当单击 ListView 控件上的 ListItem 对象标签时将触发 ItemClick 事件，该事件决定了单击的是哪个 ListItem 对象。当单击控件上的标题栏时，Column 事件将发生。

注意：这里要注意 ItemClick 事件与 Click 事件的区别，ItemClick 事件在 Click 事件之前发生。另外，单击控件的任何位置都会触发 Click 事件，而只有单击 ListItem 对象的文本或图像时，ItemClick 事件才会发生。

下面将通过一个实例来介绍控件的使用方法。程序使用 ListView 控件来显示当天蔬菜价格，单击控件的表头标题时，控件中的列表项将按照该列数据的升序排列。

（1）启动 Excel 2010，创建工作表。工作表显示当天采集的蔬菜价格。工作表的样式如图 15.67 所示。

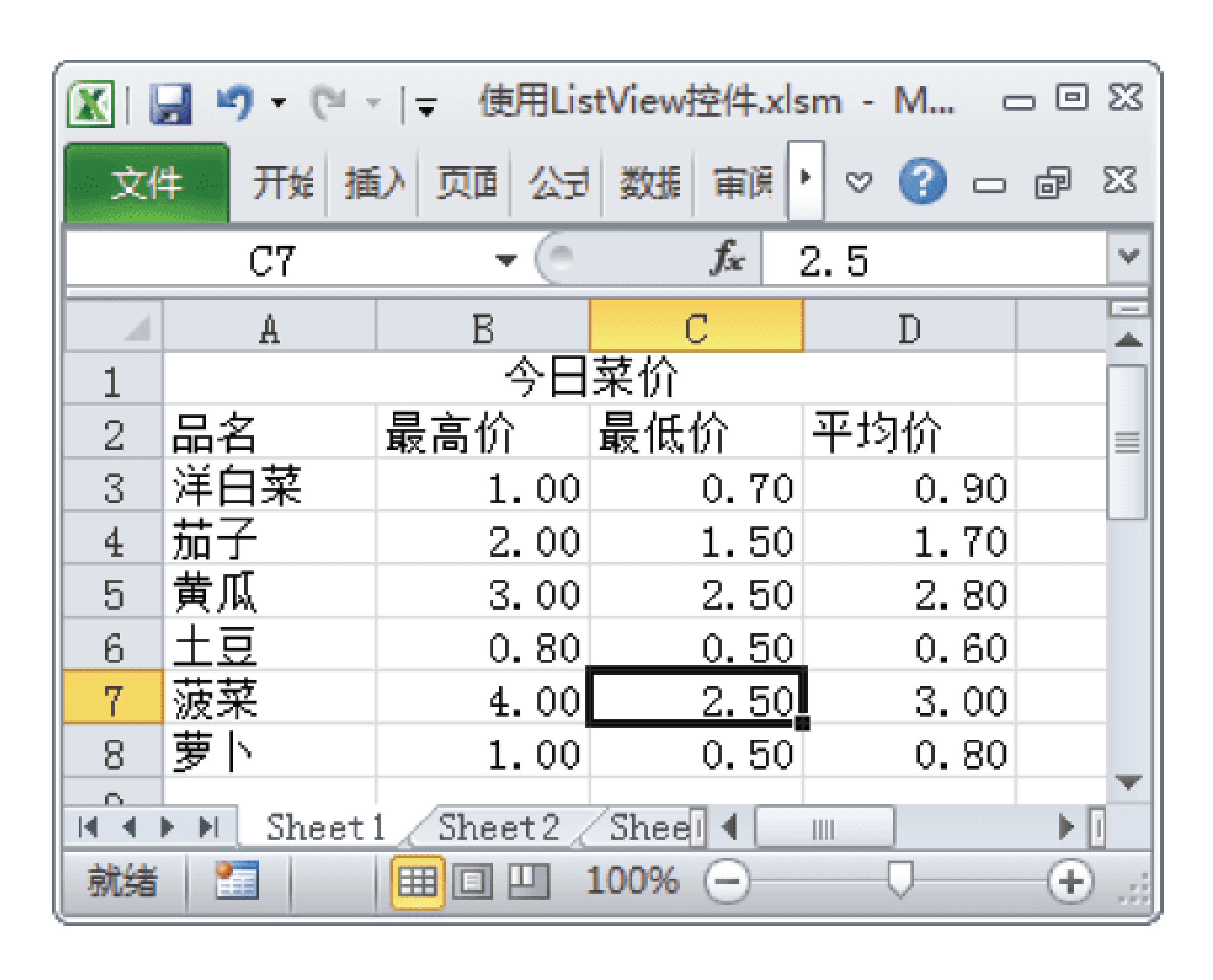


图 15.67　打开工作表

（2）打开 Visual Basic 编辑器，添加一个用户窗体，将窗体的 Caption 属性设置为“今日菜价”。在控件工具箱上右击，在弹出的快捷菜单中选择“附加控件”命令，打开“附加控件”对话框。在对话框中选择 ListView 控件，如图 15.68 所示。单击“确定”按钮，关闭“附加控件”对话框，选择的控件即被添加到控件工具箱中，如图 15.69 所示。

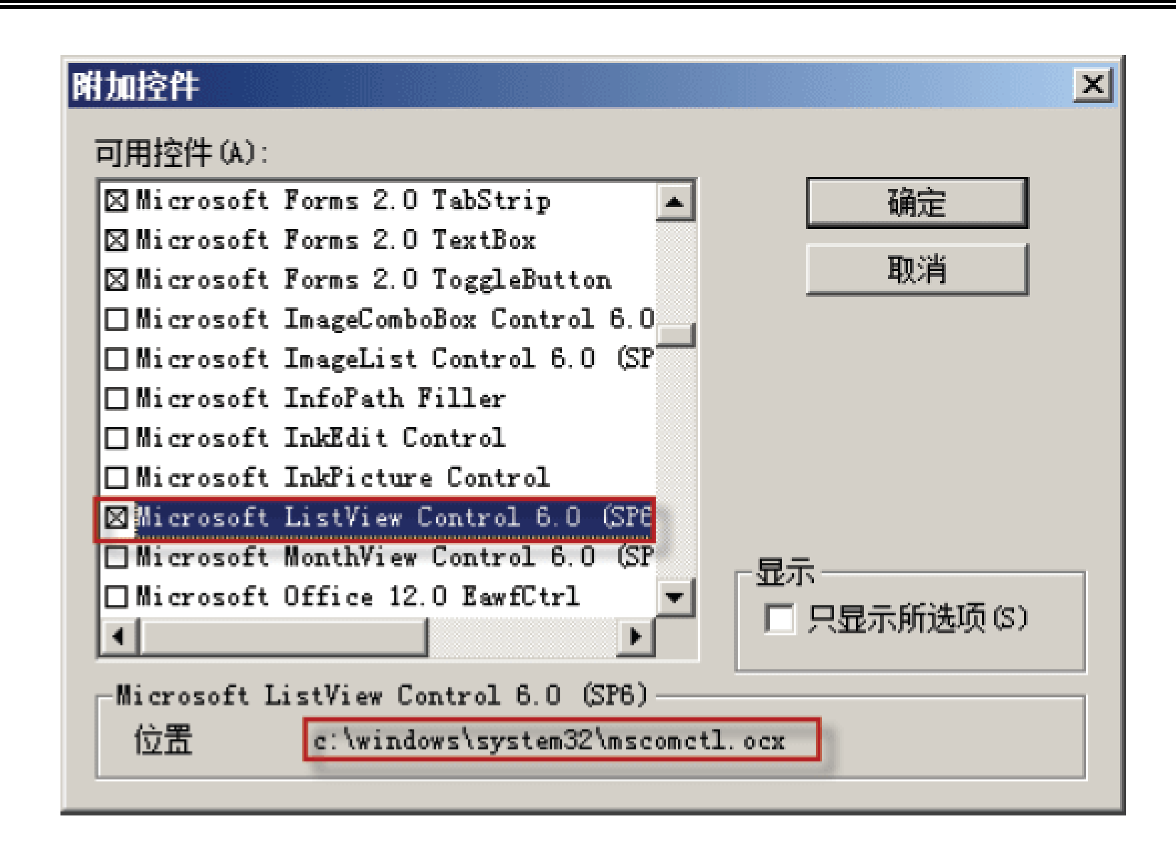


图 15.68　选择 ListView 控件

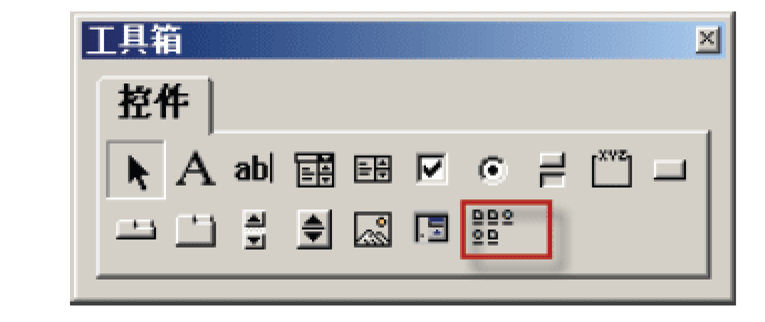
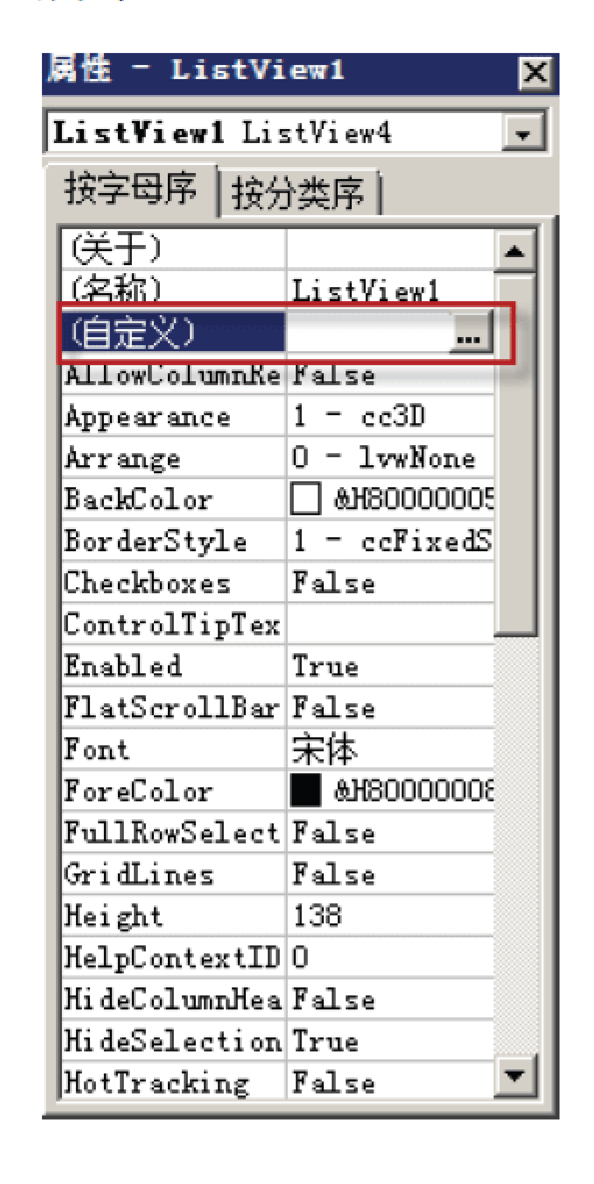


图 15.69　控件被添加到控件工具箱中

提示：若 Visual Basic 不支持该控件，可通过选择“开始”→“运行”命令，打开命令行窗口，输入命令“regsvr32 c\:windows\system32\mscomctl.ocx”向操作系统注册该控件，其中字符串“c\:windows\system32\mscomctl.ocx”表示控件所在的目录，如图 15.68 所示。

（3）在控件工具箱中选择 ListView 控件，在用户窗体中拖动光标添加控件。打开控件的“属性”对话框，选择“自定义”选项，单击出现的按钮...，如图 15.70 所示。此时将打开“属性页”对话框，选中 General 选项卡中的 Checkboxes 和 GridLines 复选框，如图 15.71 所示。



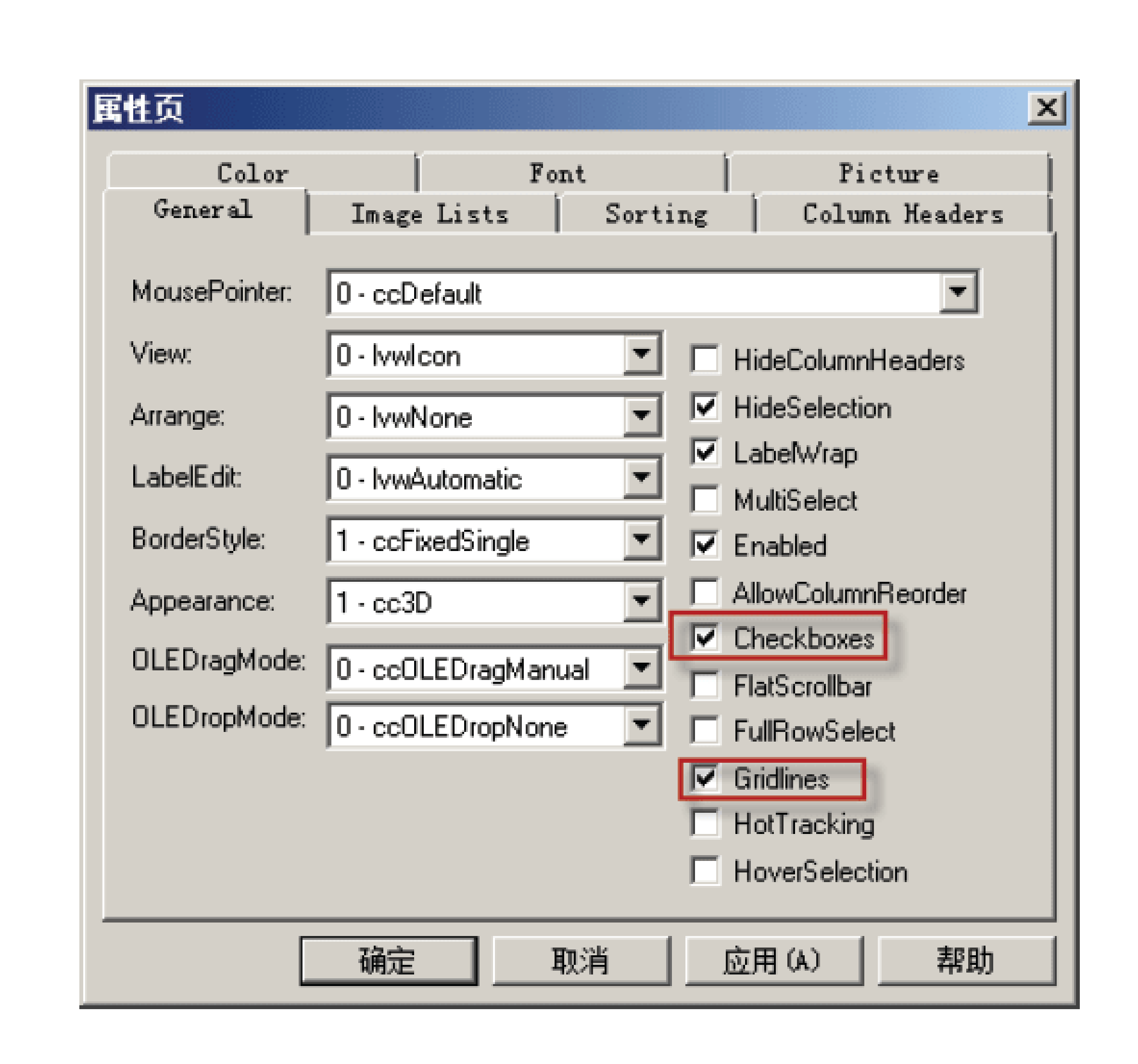


图 15.70　“自定义”选项

图 15.71　“属性页”对话框

提示：控件的“属性页”对话框给控件属性设置提供了一种可视化的方法，这种方法比直接使用“属性”对话框更加直观方便。对话框按照设置项进行分类，如，General 选项卡设置控件的常用属性，Sorting 选项卡用于设置控件显示项的排序情况，Column Headers 选项卡用于设置报表视图中的列标题情况。

🔔注意：若设置属性失败，则可以将 Msstkprp.dll 文件从 Visual Basic CD 上的 \OS\System 文件夹复制到 Windows\System 文件夹（或 System32 文件夹）。接着，使用 RegSvr32 实用程序手动注册 Msstkprp.dll。在 Windows 开始菜单上选择“运行”命令，然后输入命令 Regsvr32 C:\Windows\System\Msstkprp.dll 即可。

（4）编写窗体的初始化事件代码。该初始化事件代码主要用于向控件添加表格中的数据，控件初始化的效果如图 15.72 所示。

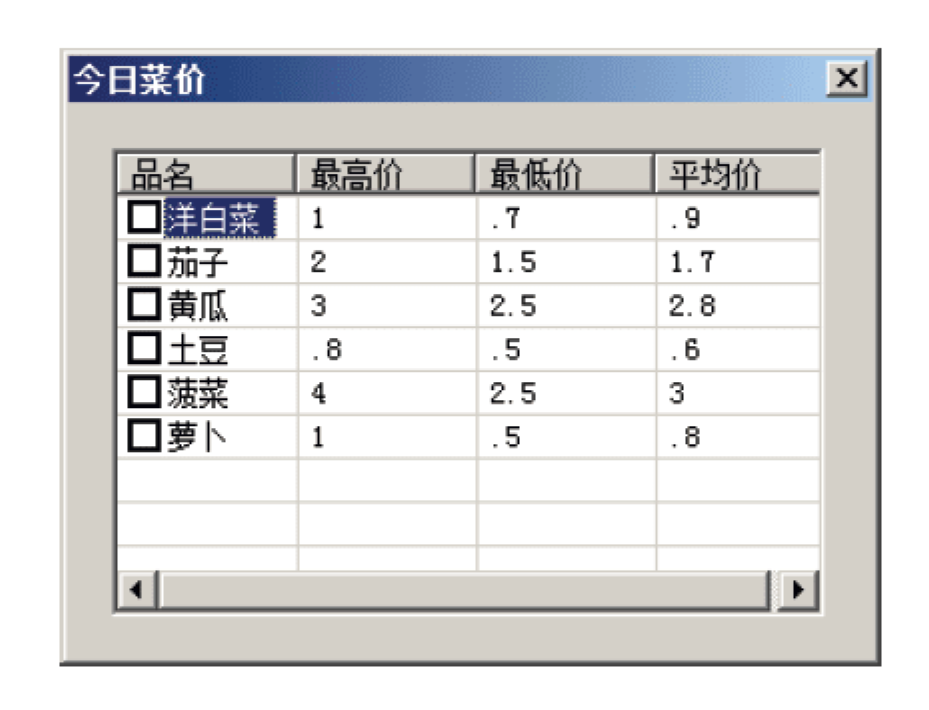


图 15.72　窗体初始化效果

窗体的详细的 Initialize 事件代码如下所示：

```
01  Private Sub UserForm_Initialize()
02      Dim n As Integer, p As Integer, q As Integer   '声明变量
03      ListView1.View = lwReport                      '设置为报表型
04      For n = 1 To 4                                 '遍历第 1 列到第 4 列
05          ListView1.ColumnHeaders.Add , , Sheet1.Cells(2, n), _
06          ListView1.Width / 4                        '添加到列标题
07      Next
08      For q = 1 To Sheet1.Cells(1048576, 1).End(xlUp).Row-2
                                                       '添加行数据
09          ListView1.ListItems.Add , , Sheet1.Cells(q + 2, 1)
                                                       '添加工作表中第一列
10          For p = 1 To 3
11              ListView1.ListItems(q).SubItems(p) = Sheet1. _
12              Cells(q + 2, p + 1)                    '其他 3 列添加为子项目
13          Next
14      Next
15  End Sub
```

【代码解析】在程序的第 05～06 行中，使用 Add 方法创建一个 ColumnHeaders 对象，即列表头对象。为了使添加的表头等宽，设置每个表头的宽度为控件宽度的 1/4。程序的第 08～09 行通过循环结构将工作表第一列的指定数据添加到控件的 ListItems 列表项集合中。而程序的第 10~13 行程序则通过循环结构将工作表其他 3 列作为子项目添加给 ListItems 集合中的每一个对象。

（5）下面为控件的表头添加鼠标单击事件。

```
Private Sub ListView1 ColumnClick(ByVal ColumnHeader As MSComctl
Lib.ColumnHeader)
    With ListView1
        .SortOrder = 0                         '按升序排列
        .SortKey = ColumnHeader.Index - 1      '指定排序的关键字
```

```
05            .Sorted = True                                  '允许排序
06        End With
07    End Sub
```

使用该事件可以在单击表头标题时，使控件中的列表项按照被单击项目的升序排列，如图 15.73 所示。

今日菜价

品名	最高价	最低价	平均价
土豆	0.8	0.5	0.6
萝卜	1	0.5	0.8
洋白菜	1	0.7	0.9
茄子	2	1.5	1.7
黄瓜	3	2.5	2.8
菠菜	4	2.5	3

图 15.73　按照“平均价”的升序排列

提示： 这里使用 ColumnClick 事件来触发事件代码，该事件在单击标题栏中的某项时发生。事件参数 ColumnHeader 表示单击项目。程序的第 02~06 行设置排序规则并允许进行排序。

15.4.2　ImageList 控件

ImageList 控件可以理解为一个图像储藏室，其用来放置窗体中需要使用的图像。在程序运行时，ImageList 控件是隐藏的，其他控件可以通过索引来调用该控件中保存的图像。

ImageList 控件包含一个 ListImages 集合，该集合由 ListImage 对象构成，该对象可以是任意大小的图像，可以被其他控件使用。ImageList 控件中支持多种常见的图像格式，包括位图（*.bmp）、光标文件（*.cur）、图标文件（*.ico）、JPEG 文件（*.jpg）和 GIF 文件（*.gif）等。

在编程时，可以通过标准的集合方法来操作 ListImage 对象，如使用 Add 方法添加对象，使用 Clear 方法清除集合对象。如，为 ImageList 控件添加一张图片，图片关键字为“Picture1”，可使用下面语句：

```
ImageList1.ListImages.Add
Key:="Picture1",Picture:=LoadPicture(ThisWorkbook.Path & "\a.bmp")
```

集合中每一个成员可以通过索引或唯一关键字来访问。当使用 Add 方法把一个 ListImage 对象添加到集合中时，索引号和唯一关键字被分别储存在控件的 Index 属性和 Key 属性中，可以通过读取属性值来直接获得。如，在载入图片时图片的 Key 值设置为“Picture1”，将该图片设置为窗体的背景，可以使用下面的语句：

```
UserForm1.Picture = ImageList1.ListImages("Picture1").Picture
```

将 ImageList 控件作为一个单一的存储库可以节约程序的开发时间，这是因为在引用图像目录时，不需要反复编写代码来加载图像。只需要依次填充 ImageList 控件，分配 Key

的值，编写使用 Key 或 Index 属性的代码来引用图像即可。

下面通过一个实例来介绍 ImageList 控件的基本使用方法。这个实例是一个美化的随机数生成器，单击窗体中的按钮将能够生成 3 位数的随机数，随机数是由数字图片构成。

（1）打开 Visual Basic 编辑器，插入一个用户窗体，将窗体的标题改为“随机数生成器”。在控件工具箱中右击，在弹出的快捷菜单中选择“附加控件”命令，打开“附加控件”对话框。在对话框中选择 ImageList 控件，如图 15.74 所示。单击“确定”按钮将控件添加到控件工具箱中。

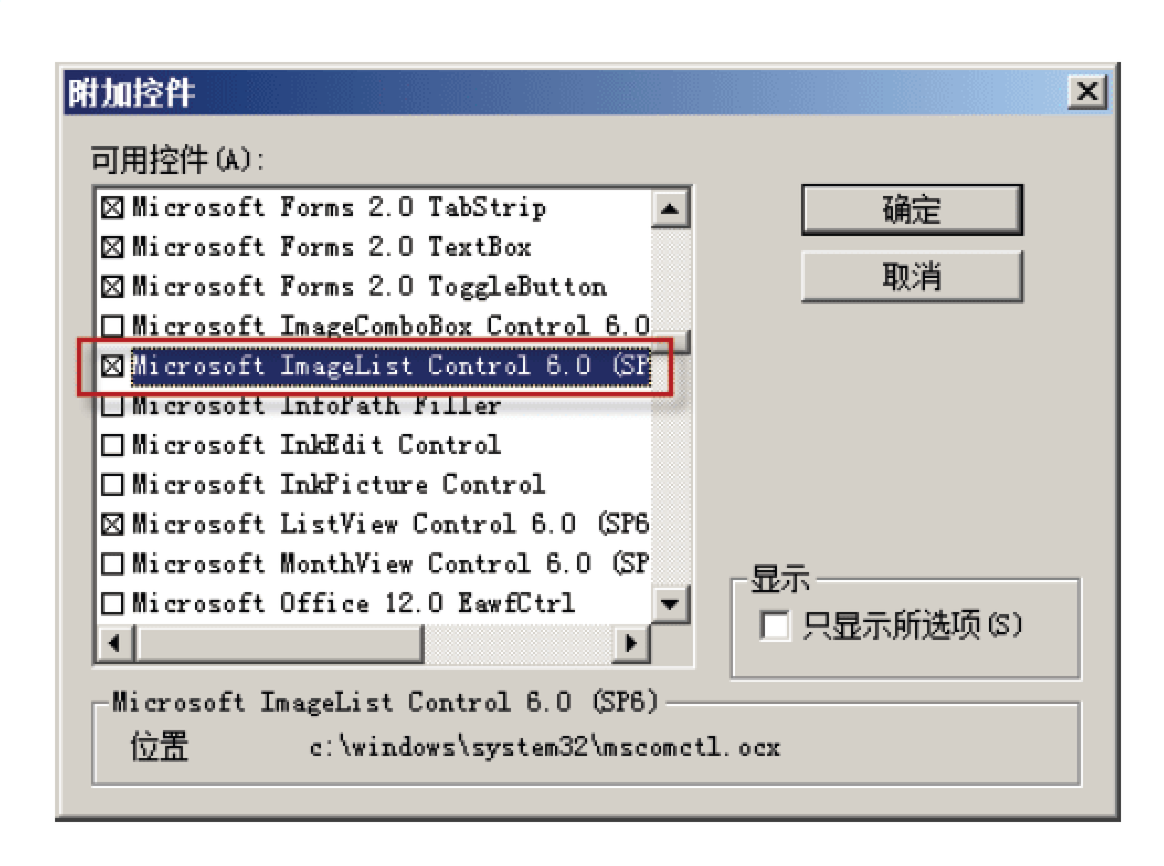


图 15.74　选择 ImageList 控件

（2）在控件工具箱中选择 ImageList 控件，在用户窗体中拖动光标创建该控件，如图 15.75 所示。再在窗体中添加 3 个“图像”控件和 1 个“命令按钮”控件。将“命令按钮”控件的 Caption 属性改为“开始”。将 3 个“图像”控件的 BorderStyle 属性设置为 fmBackStyelOpaque，即背景不透明。其他控件属性使用默认值即可，此时的窗体中控件布局效果如图 15.76 所示。

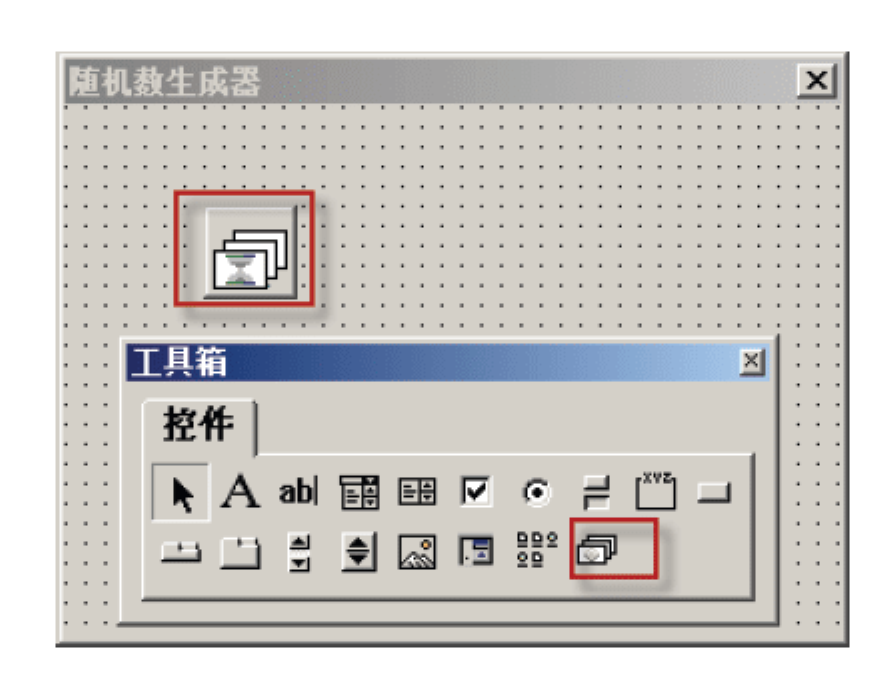


图 15.75　添加 ImageList 控件

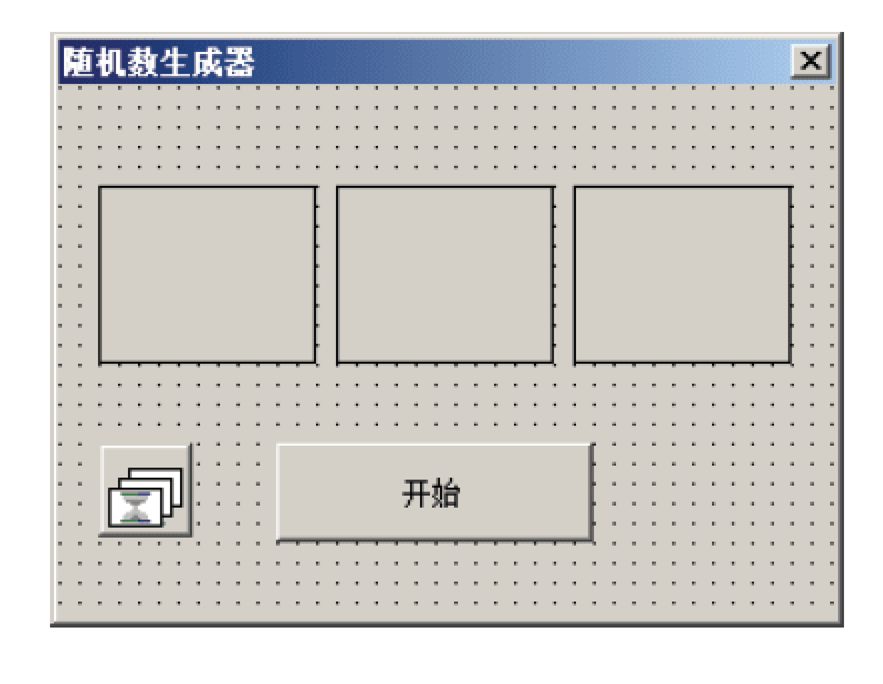


图 15.76　窗体中控件的布局

注意：ImageList 控件在程序运行时是不可见的，所以其可以在窗体中任意放置。

（3）双击窗体打开“代码”编辑窗口，在窗口中添加窗体 Initialize 事件。这里窗体的初始化事件代码将向 ImageList 控件添加图片，向 3 个“图像”控件载入图片。

```
Private Sub UserForm_Initialize()
    ImageList1.ListImages.Add Key:="i1", _
    Picture:=LoadPicture(ThisWorkbook.Path & "\1.JPG")
```

```
                                                      '向控件添加第一张图片
04        ImageList1.ListImages.Add Key:="i2", _
05        Picture:=LoadPicture(ThisWorkbook.Path & "\2.JPG")
                                                      '向控件添加第二张图片
06        ImageList1.ListImages.Add Key:="i3", _
07        Picture:=LoadPicture(ThisWorkbook.Path & "\3.JPG")
                                                      '向控件添加第三张图片
08        ImageList1.ListImages.Add Key:="i4", _
09        Picture:=LoadPicture(ThisWorkbook.Path & "\4.JPG")
                                                      '向控件添加第四张图片
10        ImageList1.ListImages.Add Key:="i5", _
11        Picture:=LoadPicture(ThisWorkbook.Path & "\5.JPG")
                                                      '向控件添加第五张图片
12        Image1.Picture = ImageList1.ListImages("i1").Picture
                                                      '"图像"控件载入第一张图片
13        Image2.Picture = ImageList1.ListImages("i2").Picture
                                                      '"图像"控件载入第二张图片
14        Image3.Picture = ImageList1.ListImages("i3").Picture
                                                      '"图像"控件载入第三张图片
15        Image1.PictureTiling = True
                                                      '设置第一张图片在控件中平铺
16        Image2.PictureTiling = True                 '设置第二张图片在控件中平铺
17        Image3.PictureTiling = True                 '设置第三张图片在控件中平铺
18    End Sub
```

窗体初始化的效果如图 15.77 所示。

提示： 这里，将图片添加到 ImageList 控件中，使用 Key 参数指定各张图片的 Key 值。在第 12～14 行代码中，3 个“图像”控件直接从 ImageList 控件中调用需要的图片。在程序的第 15 行中，将控件的 PictureTiling 属性设置为 True，允许在控件中将图片平铺。

（4）为窗体中的“开始”按钮添加 Click 事件代码。当单击“开始”按钮时，事件发生。该事件代码将生成 3 个随机数，根据生成随机数的值调用 ImageList 控件中的图像，并在窗体中显示出来，如图 15.78 所示。



图 15.77　窗体初始化的效果



图 15.78　窗体中生成的随机数效果

“开始”按钮的事件代码如下所示：

```
01    Private Sub CommandButton1_Click()
02        Dim a As String, b As Integer, c As Integer    '声明变量
03        Randomize                                      '更新随机数生成器
04        a = Int((5 - 1 + 1) * Rnd + 1)               '生成 1 至 5 之间的随机数
```

```
05        b = Int((5 - 1 + 1) * Rnd + 1)
06        c = Int((5 - 1 + 1) * Rnd + 1)
07        Image1.Picture = ImageList1. _
08        ListImages("i" & a).Picture          '“图像”控件载入变量 a 指定的图像
09        Image2.Picture = ImageList1. _
10        ListImages("i" & b).Picture          '“图像”控件载入变量 b 指定的图像
11        Image3.Picture = ImageList1. _
12        ListImages("i" & c).Picture          '“图像”控件载入变量 c 指定的图像
13    End Sub
```

提示：这里，使用 Rnd 函数分别生成 3 个随机数，以随机数来指定 ImageList 控件中的对应图像。在本段程序中应注意两点，一个是生成指定数字段间的随机数的算法，算法可参考语句 04 行。一个是注意使用变量调用 ImageList 控件中图像的方法，如语句第 08 行所示。

15.4.3　TreeView 控件

TreeView 控件可以用来显示具有层次结构的数据，如，常见的磁盘的目录结构、组织树和索引项等。这种分层结构是一个分层列表，形象地说就是一种树形结构列表。每个列表项就是一个 Node（即节点）对象，这些节点是层叠的具有分支的，同时每个节点可以包括一个图像和一个标签。大家熟悉的 Windows 的资源管理器的文件夹列表就是这种树形结构最好的体现，如图 15.79 所示。

TreeView 控件中，节点的样式可以改变。控件的 Style 属性决定了各个 Node 对象显示的图像、文本和线条的类型，该参数的值为数字 1 至 7。如，当其值为 0 时，仅为文本。其值为 1 时，为图像和文本。其值为 2，为“+/-”号和文本。其默认值为 7，即为直线、“+/-”号、图像和文本。

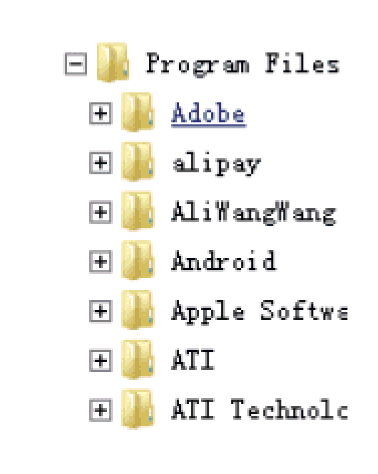


图 15.79　文件夹的树形结构

在使用 TreeView 控件时，如果需要 Node 对象的根节点或字节按字母排列，可以将控件的 Sorted 值设置为 True，此时 Node 对象根据其 Text 属性按字母顺序来排列。这里，其 Text 属性由数字开始的 Node 对象也作为字符串排列，第一个数字确定在排序中的初始位置，后面的数字确定后面的排序。如果设置为 False，则 Node 对象不排序。

提示：Sorted 属性有两种用法，在控件的顶层排列 Node 对象或对任何单独的 Node 对象的子节点排序。当其值为 True 时，仅对当前的 Node 对象排序。在控件中添加了新的 Node 对象后，该属性必须重新设置为 True 才能对添加的 Node 对象排序。

在 TreeView 控件中，常常使用 Add 方法向 Nodes 集合添加 Node 对象，其语法格式如下：

```
对象.Add (relative,relationship,key,text,image,selectedimage)
```

参数说明：

- relative 参数是可选的，代表已存在的 Node 对象的索引号或键值。

- relationship 是可选参数，代表新节点已存在节点间的关系。参数指明了 Node 对象的相对位置关系，如，当其值为 tvwFirst（即为 0）时，Node 对象和 relative 参数指定的节点位于同一层，且位于所有同层节点之前。当其值为 tvwLast（即为 1）时，该 Node 对象和 relative 参数指定的节点位于同一层，且位于所有同层节点之后。
- key 是唯一的字符串，用于在集合中查找 Node 对象，此参数可省略。
- text 参数表示 Node 对象显示的字符串。
- image 参数代表一个图像或在 ImageList 控件中的图像索引，此参数可省略。
- selectedimage 参数代表一个图像或 ImageList 控件中的图像索引，该图像在 Node 对象被选中时显示，此参数可省略。

在 TreeView 控件中，使用 GetvisibleCount 方法能够获得控件显示区域中 Node 对象的数量。Node 对象的个数取决于窗口中行数，总的行数取决于控件的高度和 Font 对象的 Size 属性。这里，可以使用 GetVisibleCount 属性来确保可视的最小行数，这样在程序运行时用户可以精确地访问一个层。如果最小行数是不可见的，可以使用 Height 属性重新设置 TreeView 控件的大小。

下面以一个实例来介绍 TreeView 控件的使用方法。本实例以树形结构显示各个方向上旅游路线的列表，项目来源于指定工作表。当选择某个项目时，下级子项目将可打开。本实例程序具有添加新项目功能，在窗口的文本框中输入项目名后单击“添加节点”按钮，该项目即被添加到选择项目的下方。在项目列表中选择某个项目，单击窗口中的“删除节点”按钮能够将选中项目删除。

（1）创建 Excel 2010 工作表，如图 15.80 所示。打开 Visual Basic 编辑器，插入一个用户窗体。在控件工具箱中右击，在弹出的快捷菜单中选择 “附加控件” 命令，然后在弹出的“附加控件”对话框中选择 TreeView 控件，如图 15.81 所示。

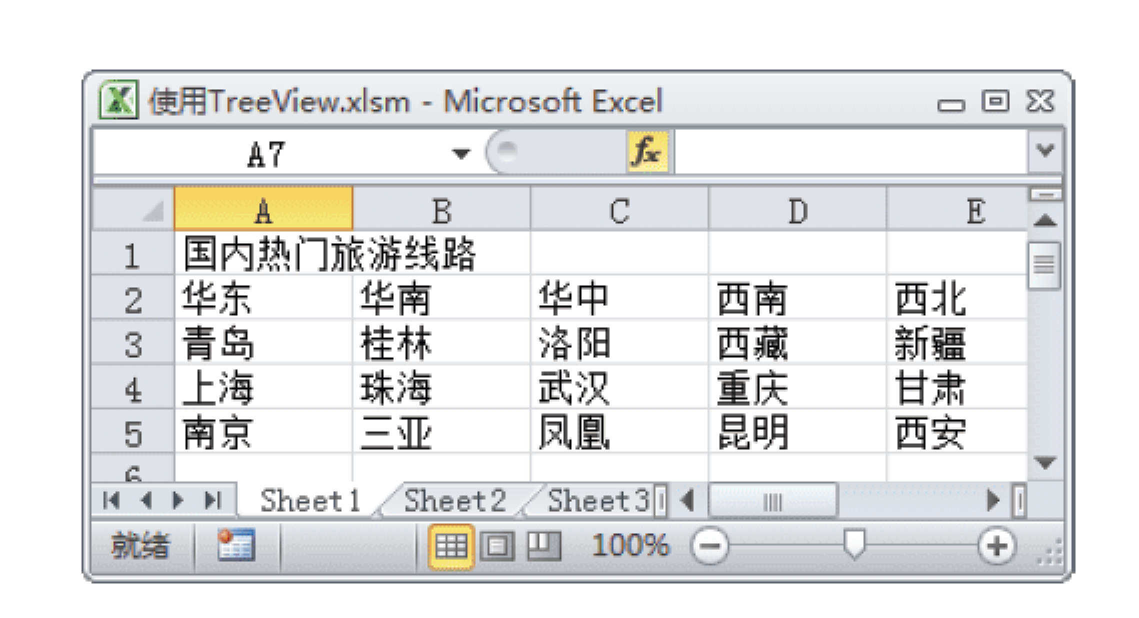


图 15.80　创建工作表

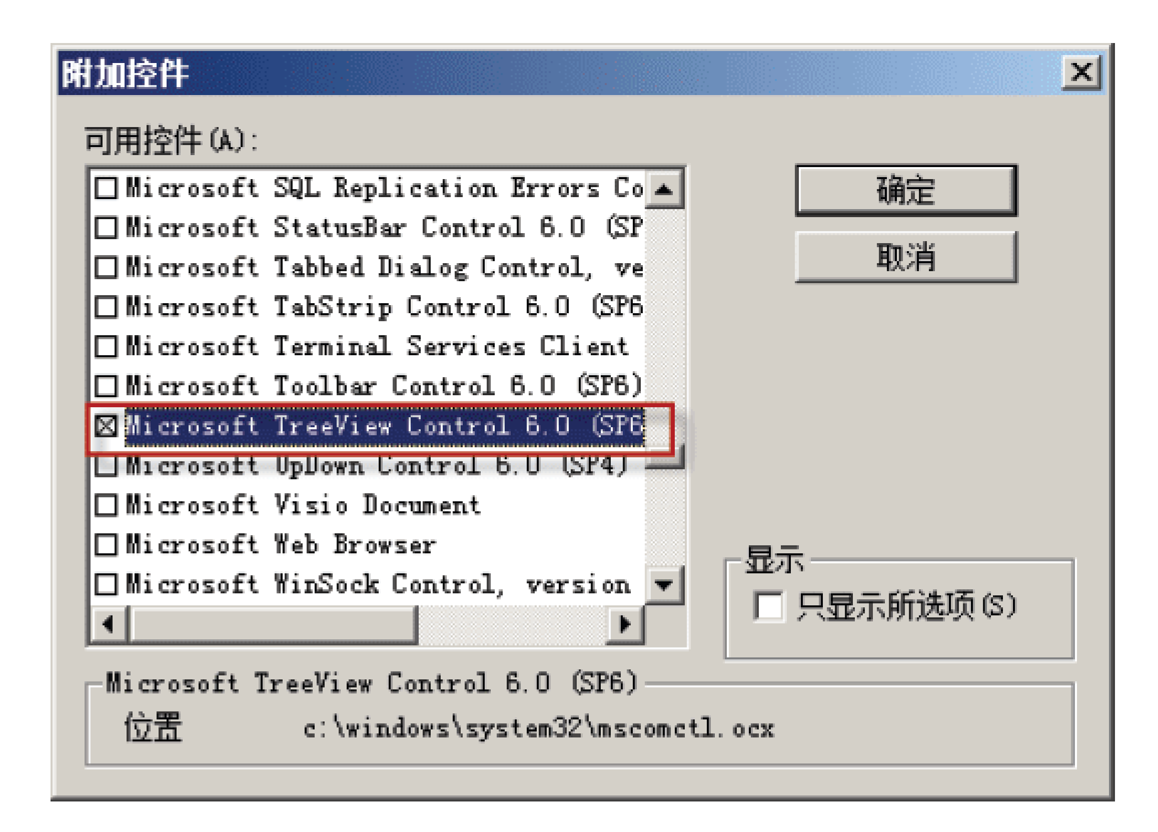


图 15.81　选择 TreeView 控件

（2）选择添加到控件工具箱中的 TreeView 控件，在窗体中拖动光标添加该控件，如图 17.82 所示。在窗体中再添加一个“文本框”控件、2 个“命令按钮”控件、1 个“标签”控件和 1 个 ImageList 控件，分别设置“命令按钮”控件、“标签”控件和用户窗体的 Caption 属性。调整窗体中控件的大小和位置，完成设置后的窗体控件布局如图 15.83 所示。

（3）为窗体添加 Initialize 事件代码。在窗体的初始化事件代码中，为 Imaglist 控件添加程序需要使用的图像，TreeView 控件读取工作表中数据得到树形结构图。窗体初始化事

件代码如下所示：

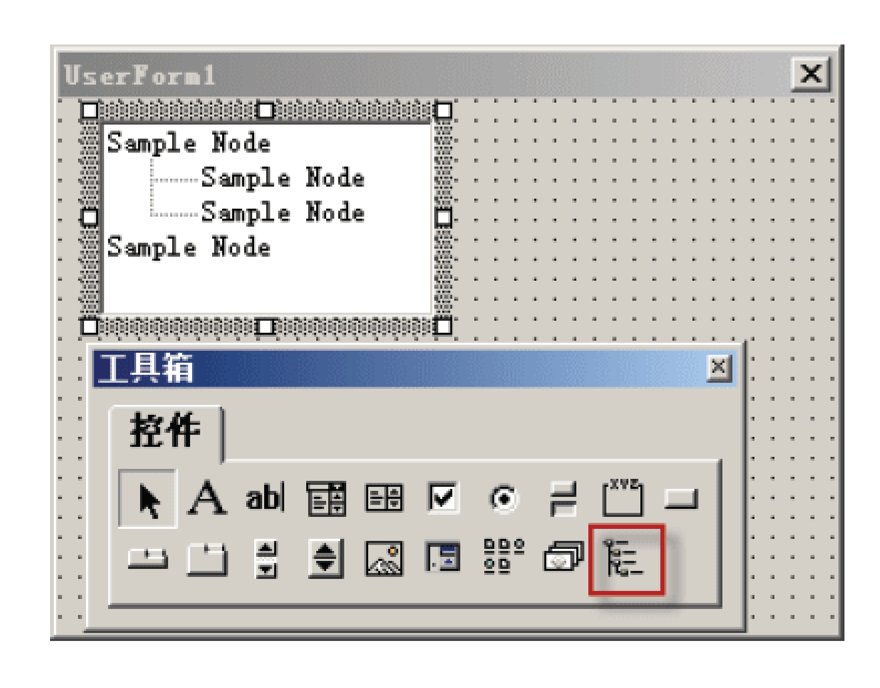




图 15.82　添加 TreeView 控件

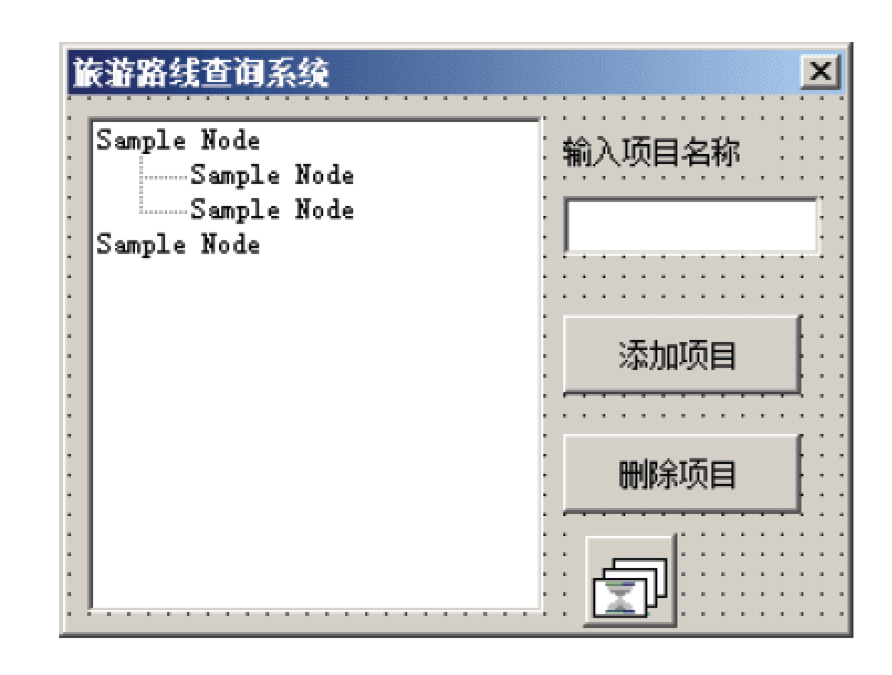


图 15.83　窗口控件布局

```
Private Sub UserForm_Initialize()
    Dim nod As Node, rng As Range                      '声明变量
    ImageList1.ListImages.Add Index:=1, _
    Picture:=LoadPicture(ThisWorkbook.Path & "\i1.bmp")
                                                     'ImageList 控件添加图片
    ImageList1.ListImages.Add Index:=2, _
    Picture:=LoadPicture(ThisWorkbook.Path & "\i2.bmp")
    ImageList1.ListImages.Add Index:=3, _
    Picture:=LoadPicture(ThisWorkbook.Path & "\i3.bmp")
    ImageList1.ListImages.Add Index:=3, _
    Picture:=LoadPicture(ThisWorkbook.Path & "\i4.bmp")
    TreeView1.ImageList = ImageList1             '与 ImageList 控件关联
    TreeView1.HotTracking = True                 '使项目能够突出显示
    TreeView1.SingleSel = True                   '展开项目时折叠其他项目
    TreeView1.LineStyle = tvwTreeLines           '设置线样式
    TreeView1.CheckBoxes = True                  '显示节点处的复选框
    Set nod = TreeView1.Nodes. _
    Add(, , "国内热门旅游线路", "国内热门旅游线路", 1, 4)
                                                 '创建根节点
    For Each rng In Range("a2:e2")               '遍历数据表第二行单元格
        Set nod = TreeView1.Nodes. _
        Add("国内热门旅游线路", tvwChild, rng, rng, 2, 4)
                                                 '创建根节点下的第一级子项
    Next
    For Each rng In Range("a3:a5")               '遍历数据表第三行单元格
        Set nod = TreeView1.Nodes.
        Add("华东", tvwChild, rng, rng, 3, 4)
                                                 '创建“华东”子项下一级子项
    Next
    For Each rng In Range("b3:b5")               '遍历数据表第四行单元格
        Set nod = TreeView1.Nodes. _
        Add("华南", tvwChild, rng, rng, 3, 4)  '创建“华南”子项下一级子项
    Next
    For Each rng In Range("c3:c5")               '遍历数据表第五行单元格
        Set nod = TreeView1.Nodes. _
        Add("华中", tvwChild, rng, rng, 3, 4)
                                                 '创建“华中”子项下一级子项
    Next
    For Each rng In Range("e3:e5")               '遍历数据表第六行单元格
        Set nod = TreeView1.Nodes. _
        Add("西南", tvwChild, rng, rng, 3, 4)
```

```
                                               '创建“西南”子项下一级子项
37          Next
38      For Each rng In Range("d3:d5")         '遍历数据表第七行单元格
39          Set nod = TreeView1.Nodes. _
40          Add("西北", tvwChild, rng, rng, 3, 4)
                                               '创建“西北”子项下一级子项
41      Next
42  End Sub
```

此时，TreeView 控件已经能够实现对象的选取，单击父项目能够展开下级的子项目，被选中项目的图标会改变，如图 15.84 所示。

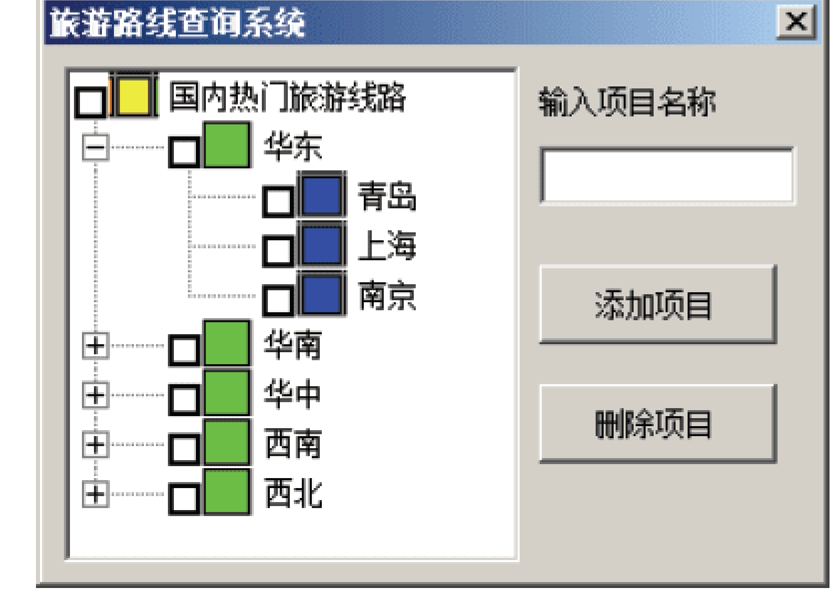


图 15.84　窗体初始化效果

提示：程序首先向 ImageList 控件添加 4 个图像文件，这 4 个图像文件，前三个用于 TreeView 控件中不同级别的项目图标，最后一个作为选择该项后的项目图标。在第 11 行将 TreeView 控件与 ImangeList 控件关联，使 TreeView 控件能够使用图片。TreeView 控件的各个项目数据来源于工作表，程序中使用 For Each… InNext 循环结构遍历单元格，将单元格中的数据添加到项目列表中。在程序的第 12~15 行设置控件的属性，这些属性也可以在“属性”对话框中进行设置。

（4）为窗体中的“添加项目”按钮添加 Click 事件代码，事件程序的流程图如图 15.85 所示。“添加项目”按钮的鼠标单击事件代码如下所示：

```
Private Sub CommandButton1_Click()
    Dim str As String, nod As Node, sel As String '声明变量
    str = TextBox1.Text                           '获得文本框文字
    If str = "" Then                              '判断是否输入
        MsgBox "请输入新项目名称！"                 '给出提示
        TextBox1.SetFocus                         '文本框获得焦点
    End If
    For Each nod In TreeView1.Nodes               '遍历所有节点
        If nod.Key = str Then                     '找到相同的项目
            MsgBox "该项目已经存在！"               '提示项目已存在
            TextBox1.SetFocus                     '文本框获得焦点
        End If
    Next
    sel = TreeView1.SelectedItem.Key              '获得子项目关键字
        If sel = "" Then                          '如果没有选择项目
            MsgBox "请选择一个项目！"               '提示选择项目
        ElseIf sel = "国内热门旅游线路" Then        '如果是根项目
            On Error GoTo check                   '错误处理
            Set nod = TreeView1.Nodes.Add(sel, tvwChild, str, str, 1)
                                                  '添加子项目
        ElseIf sel = "华中" Or sel = "华南" Or sel = "华东" _
        Or sel = "西南" Or sel = "西北" Then        '如果是第二级项目
            On Error GoTo check '错误处理
            Set nod = TreeView1.Nodes.Add(sel, tvwChild, str, str, 2)
                                                  '添加子项目
```

```
            Else
                On Error GoTo check '错误处理
                Set nod = TreeView1.Nodes.Add(sel, tvwLast, str, str, 3)
                                                              '添加第三级的子项目
            End If
        Exit Sub                                              '退出过程
        check:
        MsgBox "程序错误，将退出！"                              '错误提示
    End Sub
```

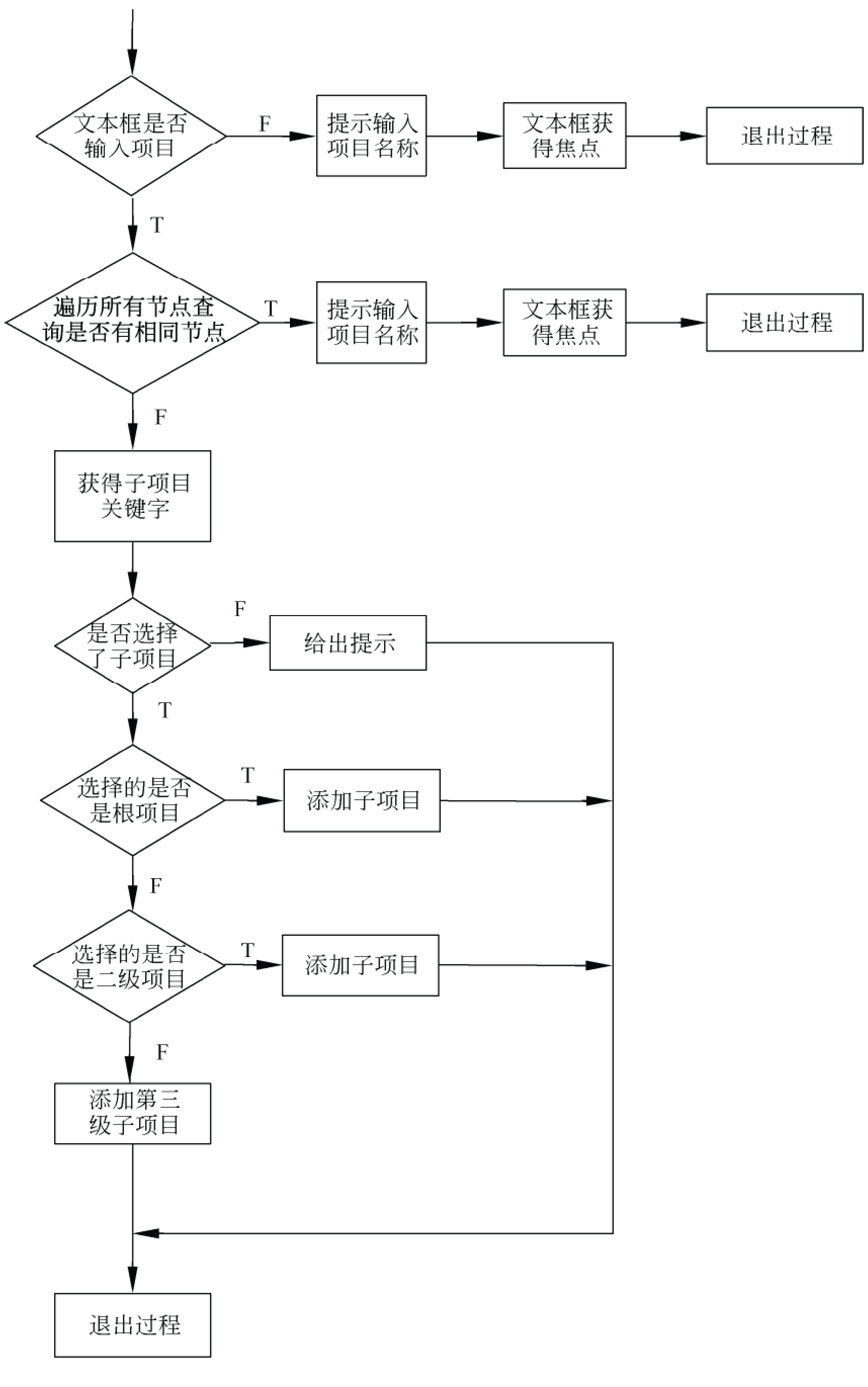


图 15.85　Click 事件程序的流程图

程序运行时，首先在 TreeView 控件中选择一个项目，然后在文本框中输入项目名称，单击“添加项目”按钮，在选择项目的下方生成一个同级的新项目，如图 15.86 所示。在单击按钮时如果没有选择项目、没有输入项目名或出现重复项目，程序将给出提示，如图 15.87 所示。

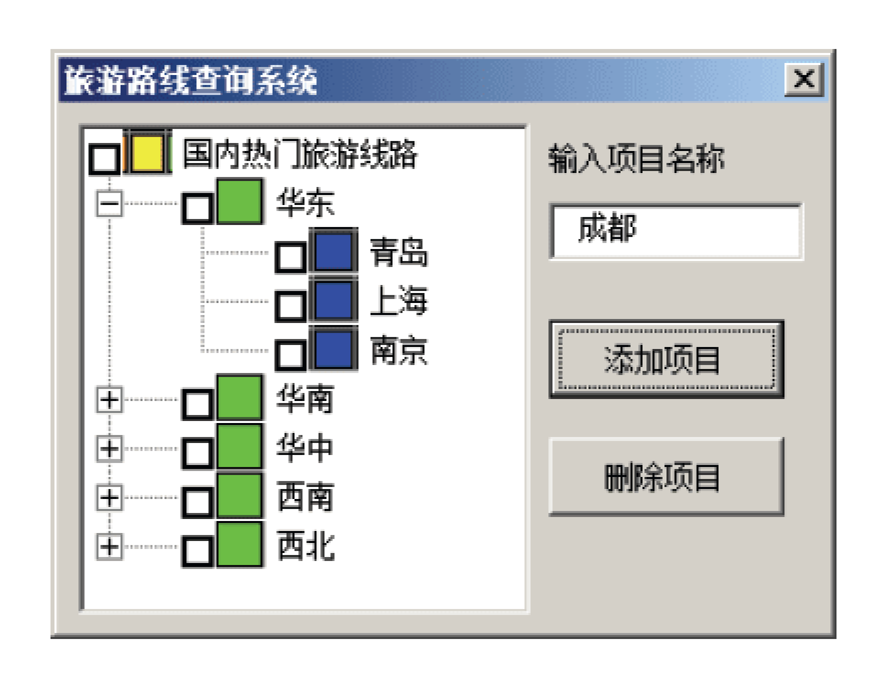


图 15.86　添加新项目

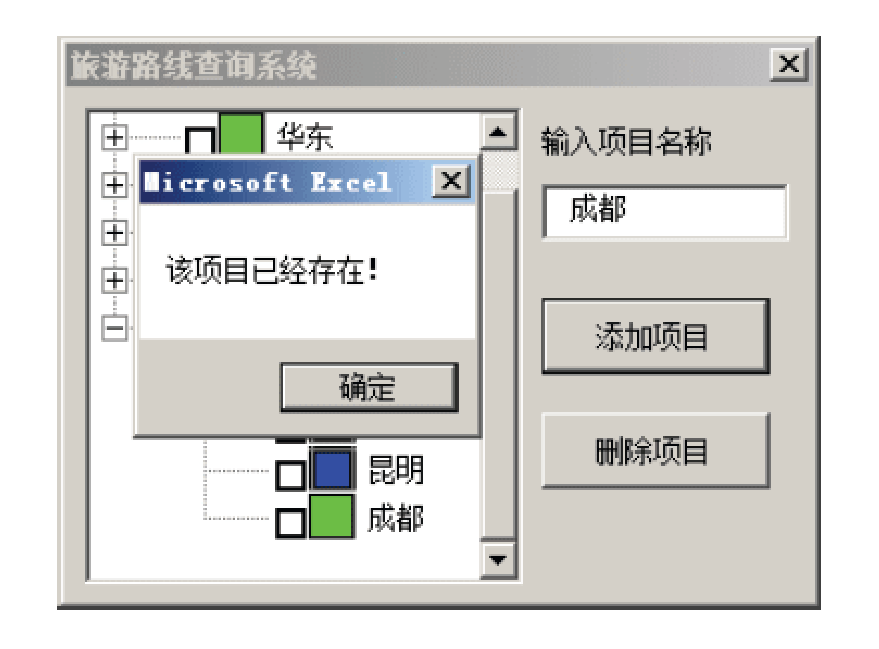


图 15.87　项目重复给出提示

提示：程序首先判断是否有项目输入，以“文本框”控件的 Text 属性值是否为空来判断。判断是否存在相同项目，是通过遍历所有的 Node 对象，以 Key 值与输入文字是否一致来判断。在第 14 行获得选择的 Node 对象的 Key 值，如果其值为空，则表示没有选择。代码的第 17 行和第 21 行判断选择项目属于哪级项目，然后分别为其添加下级子项目。之所以要分级判断主要原因是不同级别项目的图标不同，如果不考虑图标因素，可以不用判断直接添加下级项目即可。

（5）为“删除项目”按钮添加鼠标单击事件代码。“删除项目”按钮的鼠标单击事件代码如下所示：

```
Private Sub CommandButton2_Click()
    If TreeView1.SelectedItem.Index <> 1 Then       '如果不是选择根项目
        TreeView1.Nodes.Remove TreeView1.SelectedItem.Index
                                                    '删除选择项目
    Else
        MsgBox "根项目不允许删除！请重新选择需要删除的项目。"
                                                    '提示选择的是根项目
    End If
End Sub
```

当单击“删除项目”按钮时，选择的项目如果不是根项目，将被删除。如果选择的是根项目，程序给出提示，该项目不允许删除，如图 15.88 所示。



图 15.88　删除根项目时的提示

提示：在程序中，TreeView1.SelectedItem.Index 语句获得控件中选择项目的索引号，当其值为 1 时，表示选择根节点，此时程序给出提示。如果其值不为 1，表示选择其他节点，使用 Remove 方法将索引号对应的节点删除。TreeViewNode 对象索引号规则是：根节点为 1，其下级节点如“华东”、“华南”、“华中”、“西南”和“西北”分别为 2、3、4、5 和 6，“华东”的下级节点索引号为 7、8，以此类推。

15.5 小　　结

用户界面的设计，离不开窗体和置于窗体中的控件。本章介绍了 VBA 中窗体控件的添加、设置和编程的方法。在 VBA 中，控件分为标准控件和附加控件。本章介绍了 Excel 2010 中常用的标准控件的使用方法，通过具体的实例，帮助读者了解常见标准控件的作用、属性的设置方法以及对这些控件编程的技巧。对于附加控件，本章抽取了 3 个典型的控件进行介绍，相信读者能结合实例掌握附加控件使用的一般方法。

学习 VBA 编程的目的是为了减轻繁琐而复杂的数据操作，提高数据分析效率。这决定了 VBA 程序不是仅仅面对设计者，还要面对有真正操作需求的普通用户。这就需要 VBA 程序具有友好的接口，这些接口屏蔽掉程序内部的运行过程，使操作者只需通过简单的操作就能方便快捷地完成复杂的工作。

VBA 操作界面的设计过程，实际上就是窗体的自定义过程，同时也是基于窗体控件的编程过程。控件编程实际上就是通过程序修改控件属性，使用方法来为控件添加对象，使用事件来激发某种操作。理解了这些，读者将能顺利实现对控件的编程，设计出功能强大、操作方便的程序界面。

15.6 本 章 习 题

1．下面哪个控件是附加控件？（　　）

A．“单选按钮”控件　　B．“复选框”控件

C．“旋转按钮”控件　　D．TreeView 控件

2．选择下面程序的运行结果。(　　)

```
Private Sub Worksheet_SelectionChange(ByVal Target As Excel.Range)
    Dim rng As Range
    Set rng=Application.Intersect(Range("A1:G1"),Target)
    If Not rng Is Nothing Then
        UserForm1.Show
    End If
End Sub
```

A．单击工作表中单元格时显示用户窗体

B．单击工作表中非空单元格时显示用户窗体

C．单击工作表中的 A1 至 G1 区域的非空单元格时显示用户窗体

D．单击工作表中 A1 至 G1 区域的填充有数字的单元格时显示用户窗体

3．在窗体中添加 1 个“组合框”控件，编写窗体的 Active 事件代码，当窗体被激活时，下面对事件程序运行结果的描述哪项是正确的？（　　）

```
Private Sub UserForm_Active()
    Sheet1.Activate
    With ComboBox1
        .RowSource="姓名"
        .ListIndex=0
    End With
End Sub
```

A．“组合框”控件选项列表为空

B．“组合框”控件列表中存在 1 项名为“姓名”的选项

C．“组合框”控件列表的选项为 Sheet1 工作表的“姓名”列的所有内容

D．“组合框”控件列表的选项为 Sheet1 工作表的“姓名”列第一个单元格的内容

4．制作单元格文本输入器。要求用输入器输入文本时，可设置文本字体、大小和加粗、倾斜等样式。在窗口中能够输入单元格地址，将文字写入指定地址的工作表单元格中。

【提示】在用户窗体中使用“文本框”控件输入文字，使用“下拉列表框”控件选择文字字体，使用“复选框”控件设置文字是否加粗、倾斜和带下划线，使用“滚动条”控件控制文字大小。另外，使用 2 个“文本框”控件作为单元格行和列地址的输入框，使用“命令按钮”控件来控制文字的写入。程序功能的实现，关键在于使用控件的事件过程设置控件的属性。

5．使用 ListView 制作一个商品信息发布系统。要求窗体中列出商品列表，并且列表附有图标。在窗口中可以选择列表显示方式和是否排序，单击列表中选项能够显示简单的商品信息。

【提示】本实例在编写程序时，使用窗体 Initialize 事件程序为 ImageList 控件载入图片、为 ListView 控件创建选项并实现 ListView 控件与 ImageList 控件的关联，同时实现其他控件的初始化。编写 2 个“单选按钮”控件的 Click 事件代码以实现更改 ListView 列表样式，编写“复选框”控件的 Click 事件代码以实现 ListView 控件列表排序。同时，使用 ListView 控件的 Click 事件来编写代码实现在“标签”控件中显示列表中选择项目的简单提示信息。

第 16 章　自定义 Excel 2010 功能区

在 Excel 2010 中，用户的交互界面与 Excel 以前的版本有所不同，其主要的功能主要集中于操作界面的选项卡上。Excel 2010 的功能界面除了系统自带的功能外，也可通过自定义功能区显示的内容，同时也可以根据需要定制自定义功能的按钮显示，同时还可以使用 UI 编辑器重新设计 Excel 2010 的功能界面。本章所要学习的内容和目标如下所述。

- ❑ 了解 XML 文件格式及其构成形式；
- ❑ 了解 Excel 2010 功能区的基本控件，掌握其常用控件的作用和 XML 标识；
- ❑ 了解 Excel 2010 功能区的基本容器，掌握其基本容器的使用方法；
- ❑ 掌握使用 XML 文件自定义功能区的一般步骤，能够使用此方法自定义功能区；
- ❑ 了解使用 UI 编辑器自定义功能区的一般步骤，能够使用 UI 编辑器定义功能区。

16.1　什么是 Open XML

要掌握 Excel 2010 功能区的定制方式，首先需要了解 XML 和 Open XML 的知识。XML 是 Extensible Markup Language（即可扩展置标语言）的缩写，这是一种可以用来创建用户自己标记的置标语言。XML 可以用来存放数据，设计它的目的是为了描述数据，即，什么是数据，如何存放数据。XML 结构简单，具有很强的通用性，易于在任何应用程序中读写数据，这使其成为数据交换的公共语言，现在越来越多的应用程序都支持 XML。

Office 2010 引入了新的文件格式，即 Open XML 格式。在 Office 2010 中，保存的文件扩展名都是以“x”结尾，如，Word 文档为 docx、Excel 工作簿文档为 xlsx，PowerPoint 演示文稿为 pptx。这种新的文档保存技术实际上使用的是两种常见的技术，一个是刚才介绍的 XML，另一个则是文件压缩的 Zip 技术。

作为一种基于文本的格式，XML 本身的压缩性就非常好了，而 Office2010 的 Open XML 借助于 ZIP 技术对文档进行保存，这样能够有效地减小 Office 文档的大小。这种方式有利于提高用户工作效率，减小磁盘存储空间，便于文档的传播交流。同时，Open XML 文件格式也使得 Office 文档由封闭转向开放，使 Office 文档与其他应用程序间的交流更为方便，文档的可靠性、安全性和易用性都得到了增强。

实际上，Office 2010 的 XML 格式的文件是由一些文档部件组成的。ZIP 技术会在文档保存时将这些部件封装在一个 ZIP 容器中，对其压缩打包后以一个单文件形式进行保存。当打开文档时，文档会自动解压缩。解压缩和压缩操作都在后台进行，用户感觉不到这种操作。为了理解 Open XML 格式文件的这个特点，不妨完成下面的实验：

（1）使用 Excel 2010 创建一个成绩表，如图 16.1 所示。将该文件保存为“成绩表.xlsm”。

（2）关闭 Excel 2010，在 Windows 资源管理器中将该文件的扩展名改为“.zip”。使

用 WinRAR 这个常用的文件解压缩软件打开该文件。此时可以看到，这个 Excel 工作表文件实际上就是一个压缩包，其中包含了目录结构和文件，如图 16.2 所示。这些文件就是 Excel 文件中的元数据、文档部件、文档属性、缩览图和图形对象等。

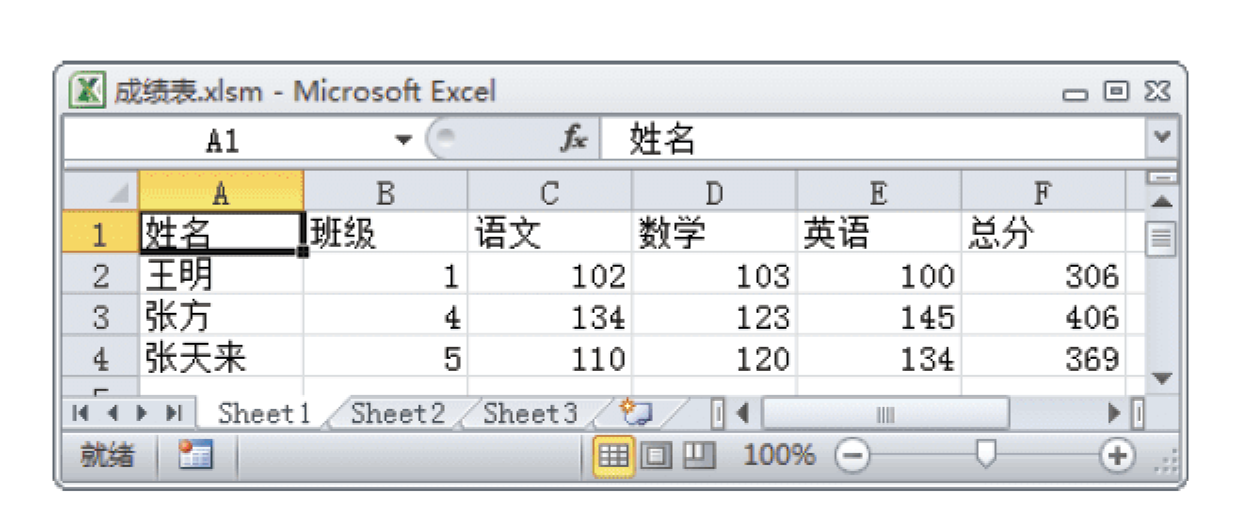


图 16.1　创建的 Excel 2010 工作表

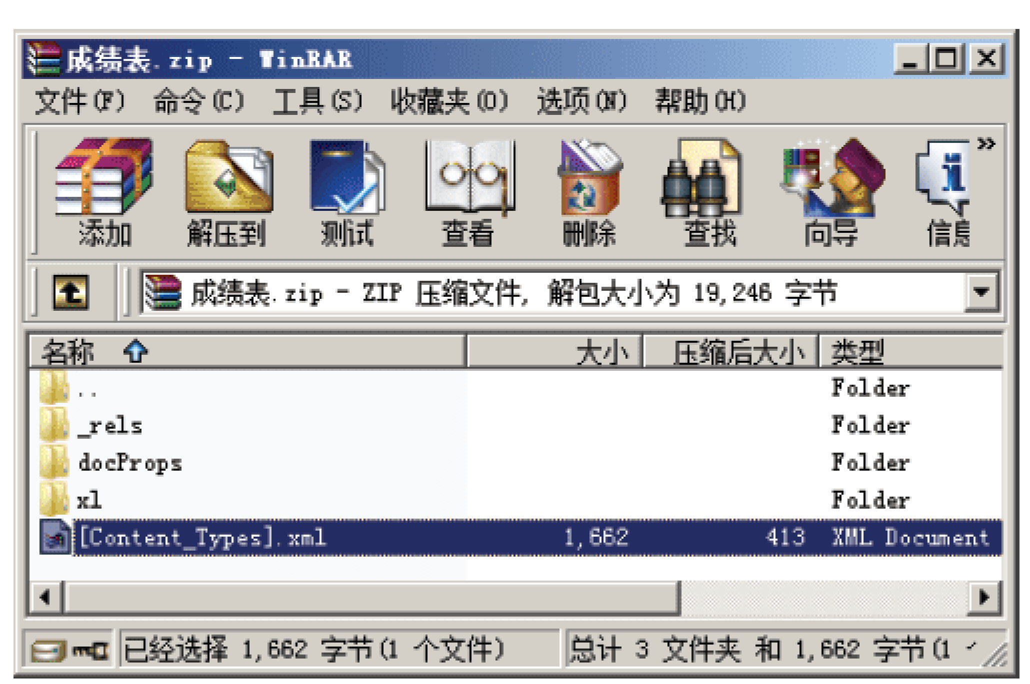


图 16.2　压缩包中的 Excel 2010 文件结构

提示： 如果没有安装 WinRAR 也没有关系，Windows XP 对 Zip 文件提供了支持。在 Windows 的资源管理器中，Zip 压缩包可以像一个文件夹那样查看其内容。

（3）双击 xl 文件夹中的“ShareString.xml”文件，在 IE 浏览器中可以查看该文件，此时可以看到该文件包含工作表中的学生姓名记录，如图 16.3 所示。



图 16.3　XML 文件中的学生姓名

16.2　了解 Excel 2010 的功能区

与早期版本相比较，Excel 2010 的变化首先体现在操作界面上。过去大家熟悉的菜单栏和工具栏被摒弃，取而代之的是全新的功能区。在功能区中，命令被组织在组中，组被集中在选项卡中，功能区中的每个选项卡都与一类操作活动相关。

Excel 2010 的功能区是可以由用户来定制的，功能区的自定义分为两种情况，一种情

况是文档级的自定义选项卡，这种自定义选项卡被绑定在特定的文档中，只能被该文档使用。另一种是程序级的自定义选项卡，其通过 Excel 加载宏来加载，所有的 Excel 文档都可以使用。下面通过一个实例来熟悉 Excel 2010 的功能区。

16.2.1　使用 Excel 2010 功能区的基本控件

从程序设计角度来看，Excel 2010 功能区实际上是由不同的控件组成的，选项卡、命令组和按钮都是控件。功能区的控件很多，主要分为基本控件和容器控件这两类。功能区的基本控件包括按钮、切换按钮、复选框、分隔条和下拉库列表。下面，通过创建一个公司年度销售业绩为例来介绍功能区的基本控件。

（1）启动 Excel 2010，在工作表中创建工作表标题、相关项目标题以及人员编号和人员姓名，如图 16.4 所示。

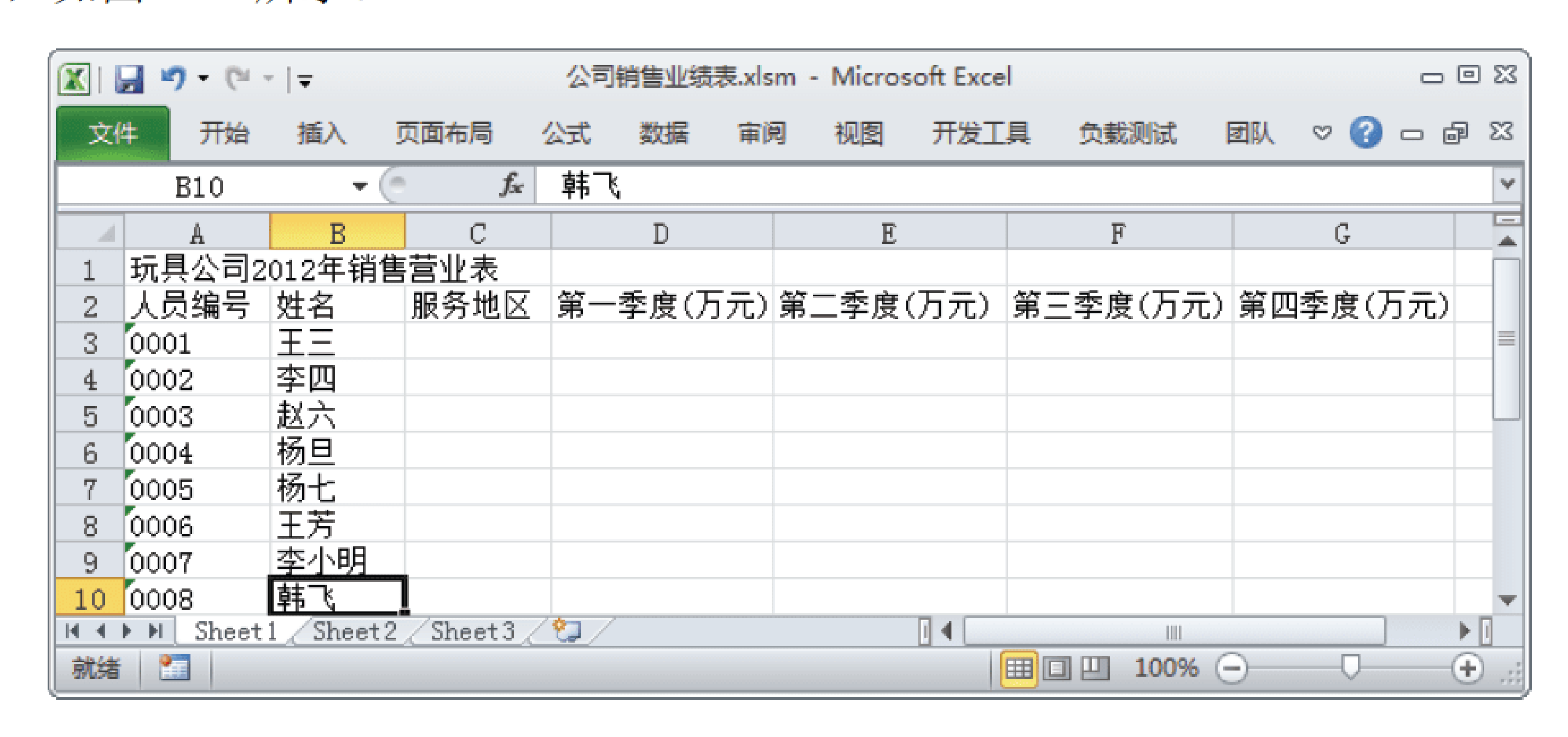


图 16.4　创建工作表

（2）在“服务地区”栏中输入第一个销售人员的服务地区，在功能区的“开始”选项卡中单击“复制”按钮。此时，选择单元格的内容被复制到剪贴板中，该单元格被蚂蚁线包围，如图 16.5 所示。在工作表中选择单元格，单击“开始”选项卡中的“粘贴”按钮，刚才复制的地区信息被复制到选择的单元格中，如图 16.6 所示。采用相同的方法完成各销售人员服务地区的填写。

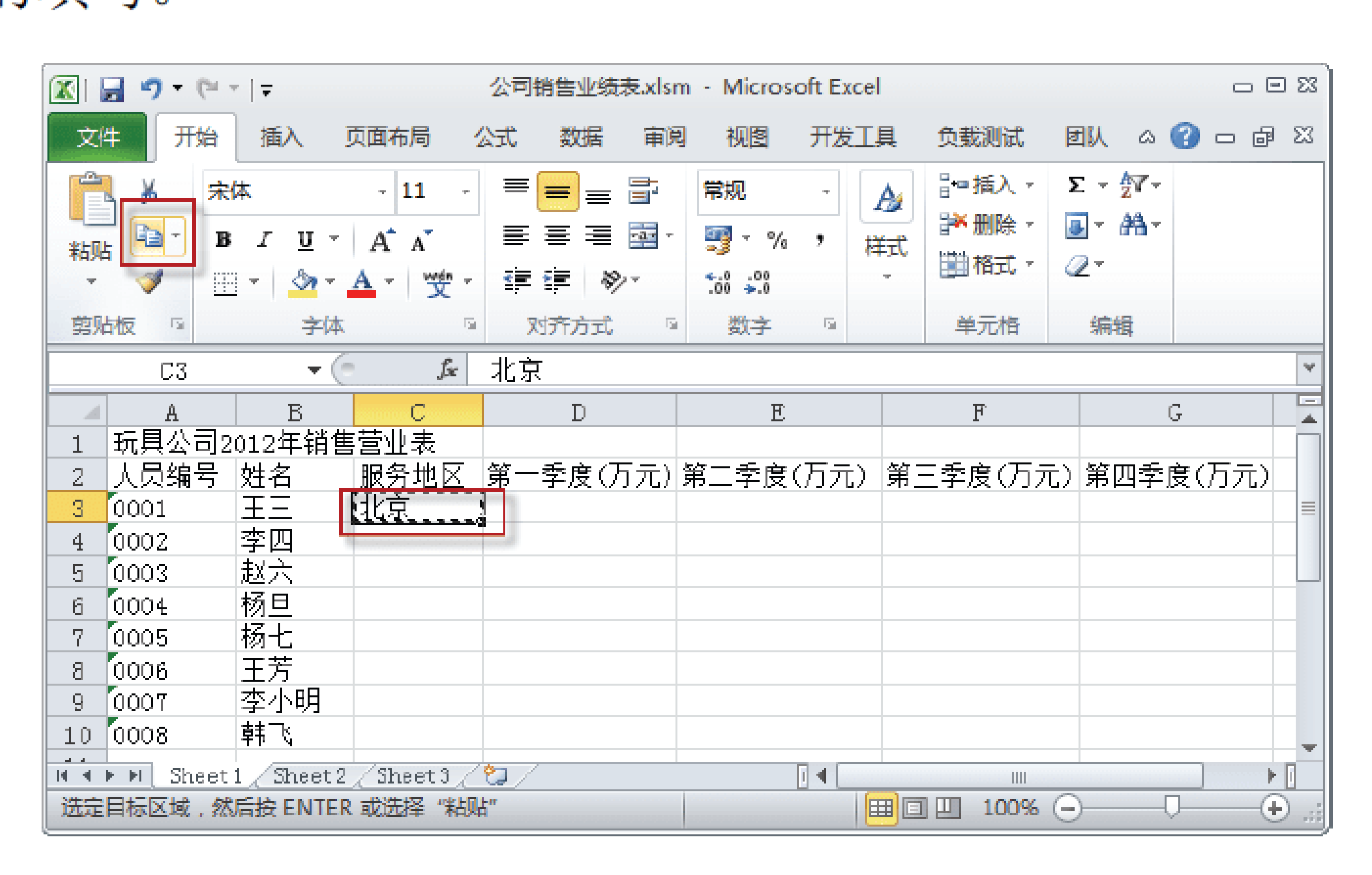


图 16.5　进行复制操作

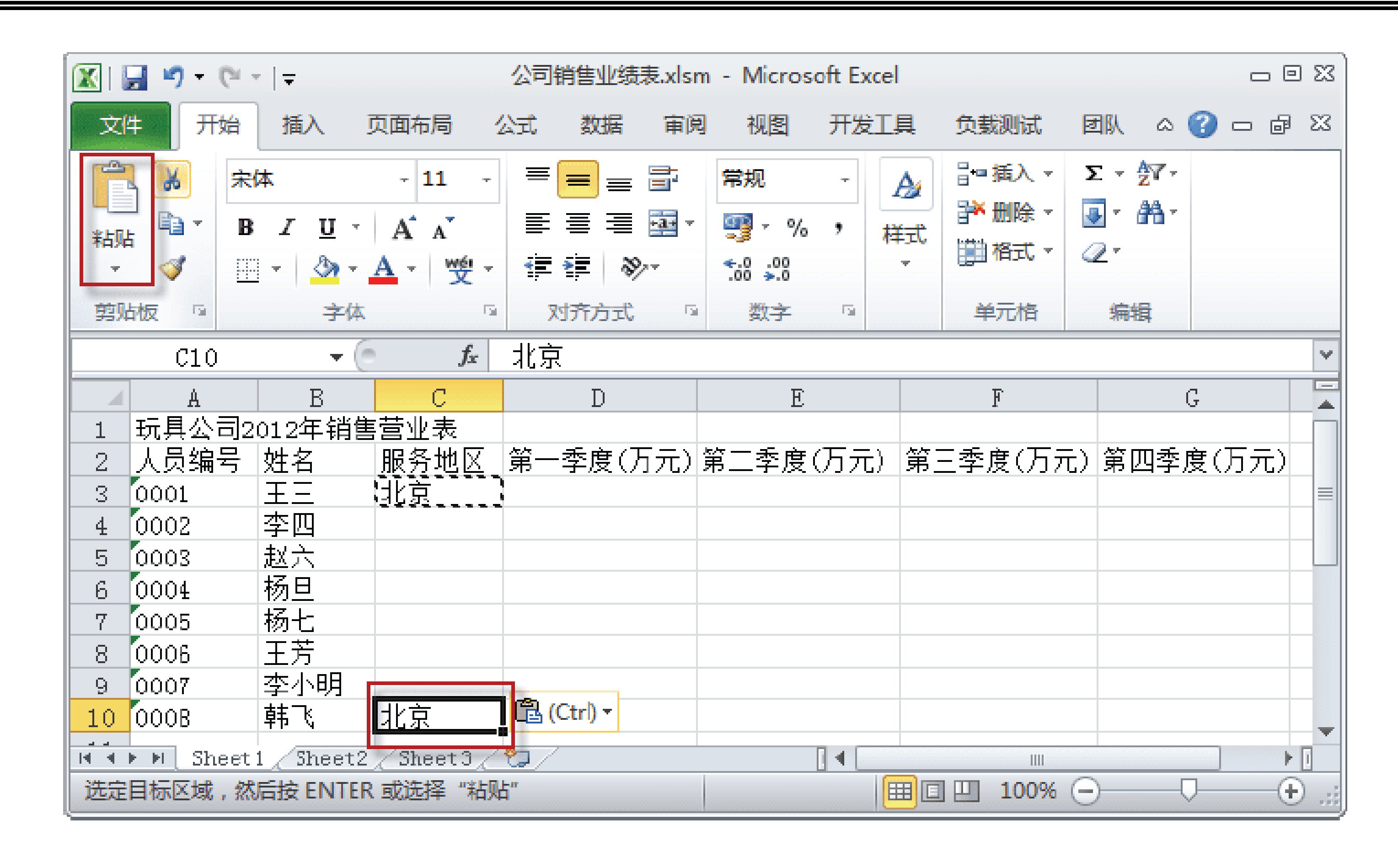


图 16.6　粘贴地区信息

提示： 按钮是功能区中最为常见的控件，如这里使用的“开始”选项卡中的“剪切”和“复制”等按钮，如图 16.5 所示。在功能区中，按钮可以带有标题和图标，能够接收用户的单击，并调用那个相应的过程来完成任务。按钮在 XML 中的标识为<button>。

（3）在工作表中输入销售信息数据，拖动光标框选输入的数据。在“开始”选项卡的“对齐方式”组中单击“居中”按钮使数据居中。同时，单击“字体”组中的“加粗”按钮使文字加粗显示，如图 16.7 所示。此时可以看到，被单击的两个按钮加亮显示，处于按下状态。如果需要取消文字的“加粗”样式，可以再次单击“加粗”按钮，取消其按下状态即可。

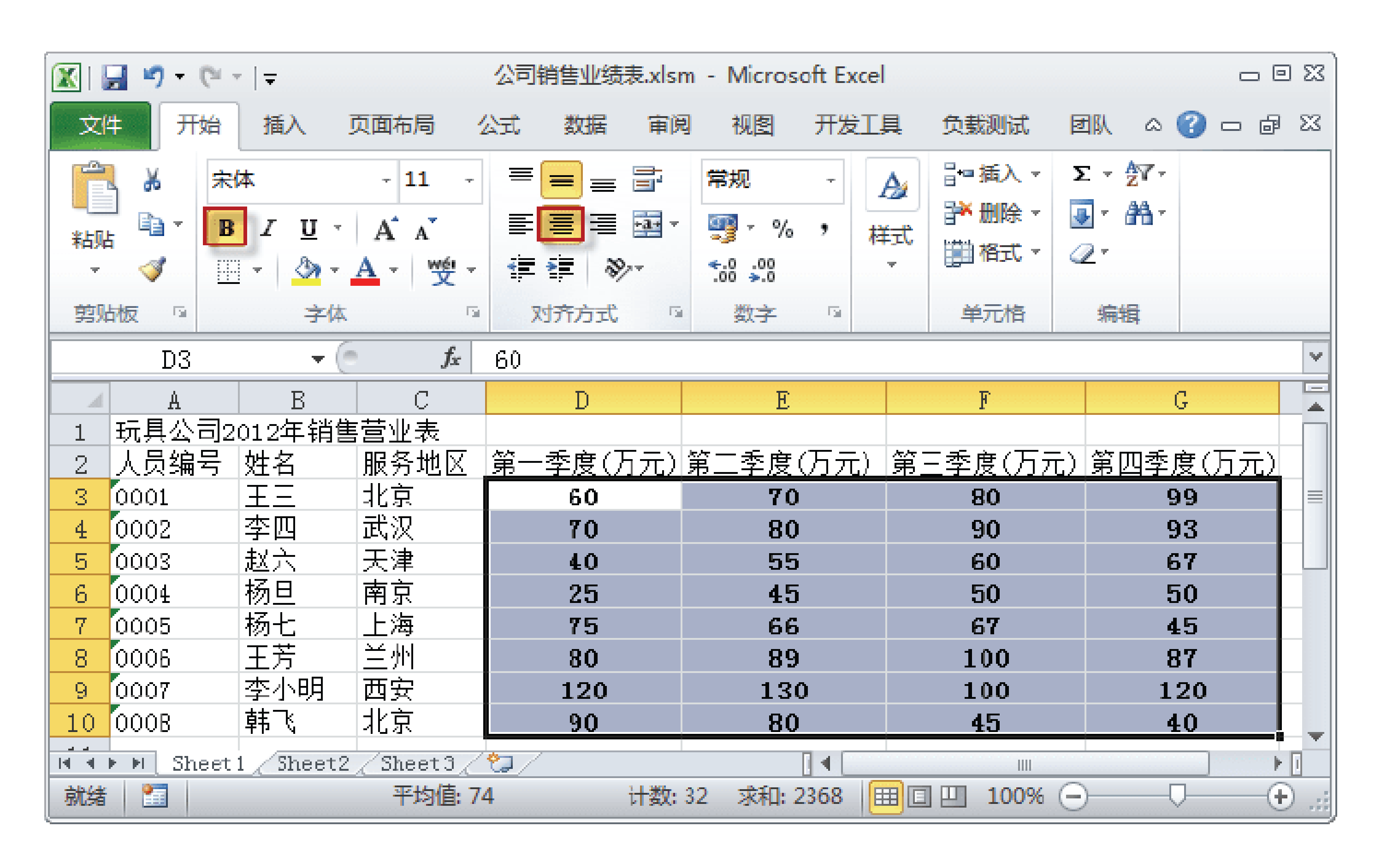


图 16.7　设置文字居中对齐和加粗

提示： 在上面使用的“加粗”按钮和“居中”按钮属于功能区的基本控件——切换按钮。切换按钮在单击时按下，再次单击变为非按下状态。该按钮常用于在两个状态之间进行切换，或在一个组中的多个可能状态中切换一个，如，“开始”选项卡的“对齐方式”组中的“文本左对齐”、“居中”和“文本右对齐”按钮。切换按钮在 XML 中的标识为<toggleButton>。

（4）在功能区中选择“页面布局”选项卡，取消对“工作表选项”组中“标题”下的“查看”复选框的选中。此时，在 Excel 2010 工作窗口中将不再显示表示行列的纵横标题，如图 16.8 所示。

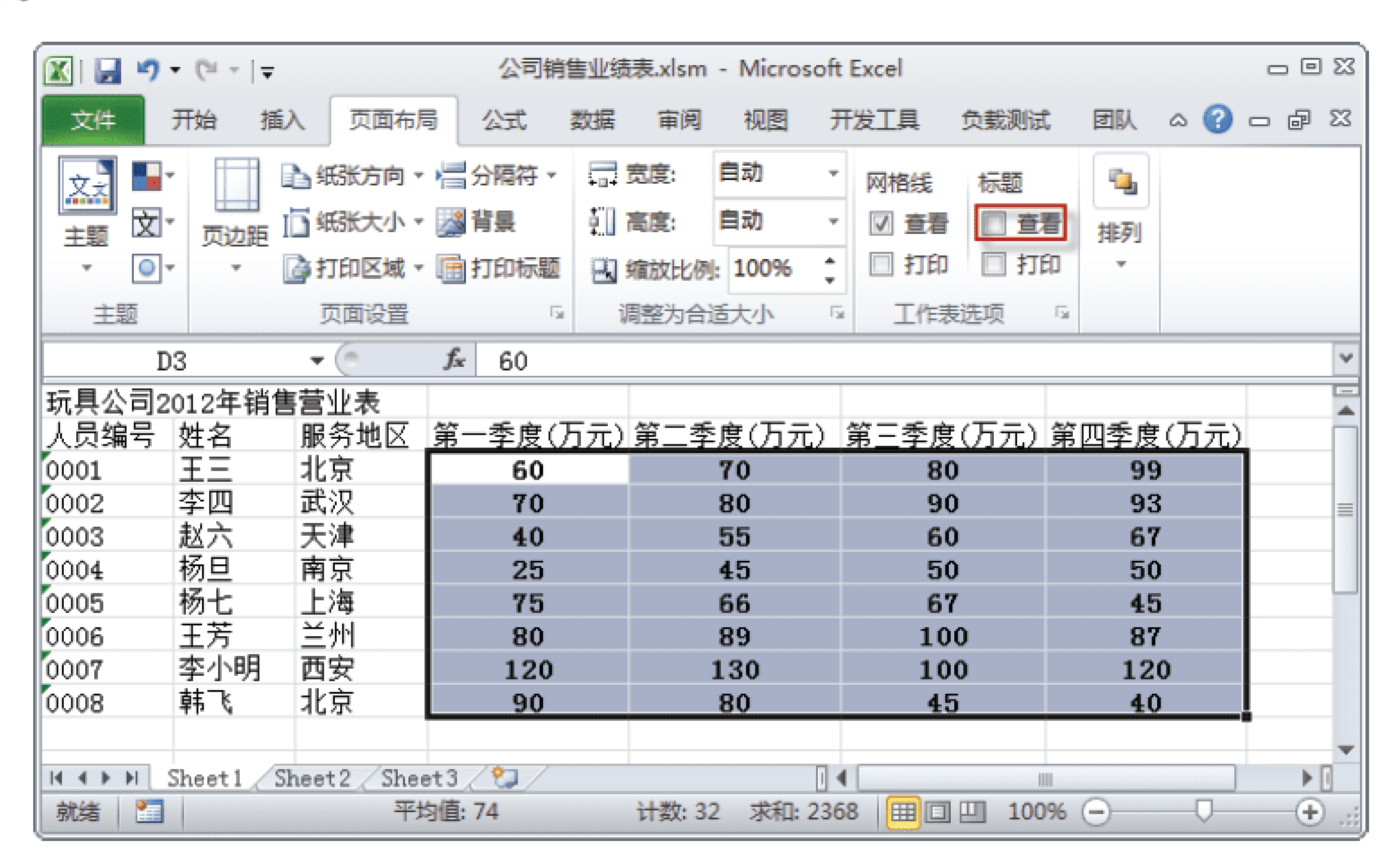


图 16.8　取消行列标题的显示

提示：功能区中的“复选框”控件是功能区的基本控件之一，其作用与第 15 章介绍的“复选框”控件相同，如 “页面布局”选项卡的“工作表选项”工作组中的“查看”复选框等。复选框在 XML 中的标识为<checkBox>。在功能区中常常可以见到分隔条，其用于在组中对控件进行分隔。如，在本步中“工作表选项”组中“网格线”与“标题”间的竖线即为分隔线，如图 16.9 所示。分隔条在 XML 中的标识为<separator>。

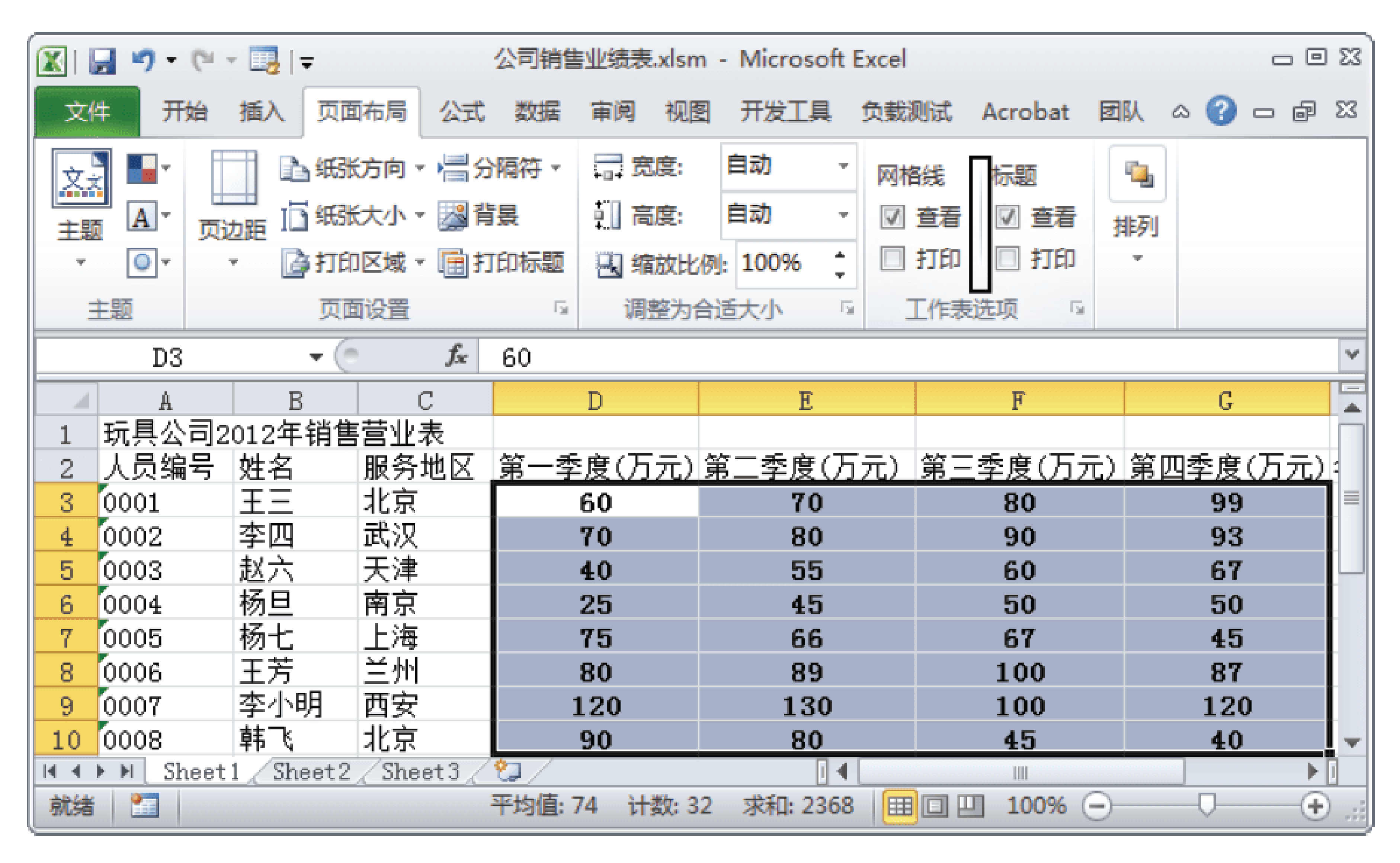


图 16.9　“网格线”与“标题”间的分隔线

（5）下面为表格的打印制作一个水印。选择“插入”选项卡，单击“艺术字”按钮，在打开的下拉列表中选择一款艺术字效果，如图 16.10 所示。在出现的艺术字文本框中输入文本，效果如图 16.11 所示。

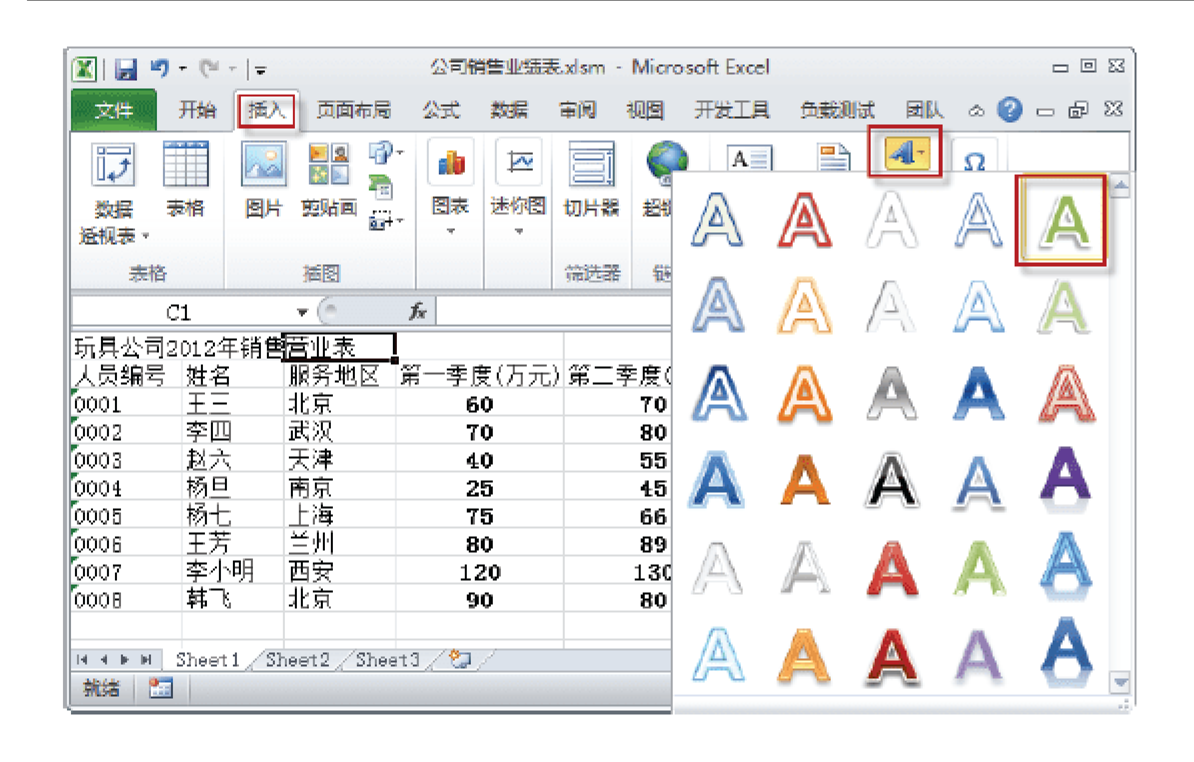


图 16.10　选择一款艺术字

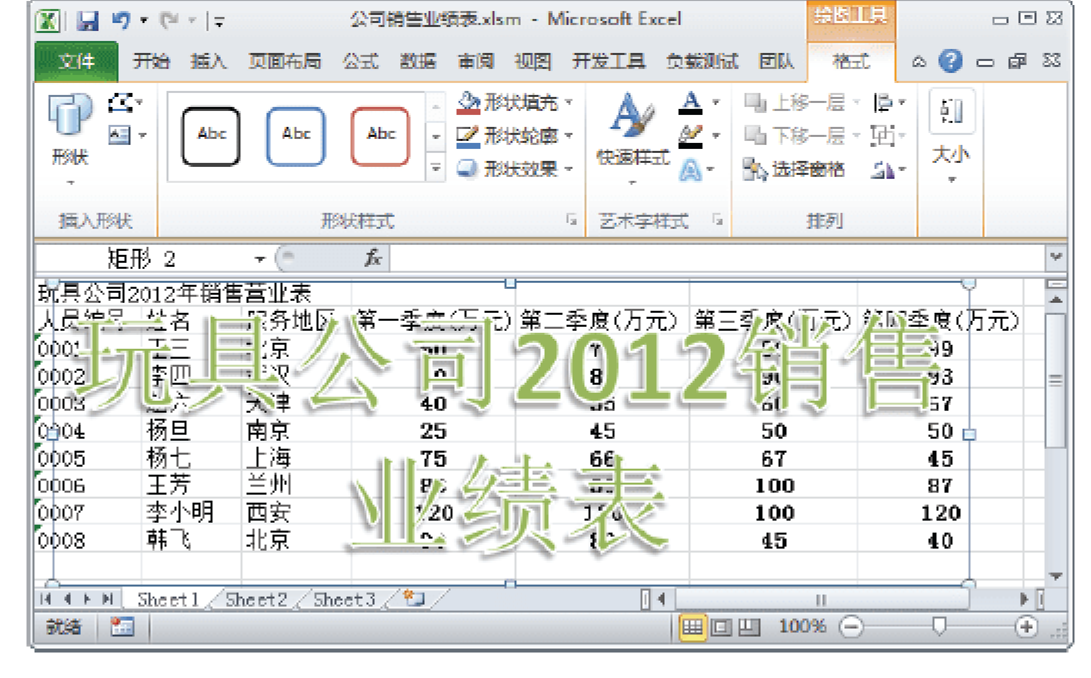


图 16.11　创建艺术字

提示： 在功能区中，下拉库列表是由下拉控件和一组其他控件构成的。在下拉库中，列出了多项系统内置的控件，单击这些控件能够直接实现某种功能。下拉库列表最为典型的就是“插入”选项卡的“文本”组中的“艺术字”下拉库列表。下拉库列表在 XML 中的标识为<gallery>。

（6）在功能区的“格式”选项卡中，单击“艺术字样式”组中的“设置文本效果格式”按钮，在打开的“设置文本效果格式”对话框中选择“文本填充”选项，将其中的“透明度”值设置为 100%，如图 16.12 所示。适当设置“文本边框”的“透明度”值，此时可以预览到文字被设置透明度后的效果，如图 16.13 所示。单击“关闭”按钮关闭对话框，完成水印的制作。

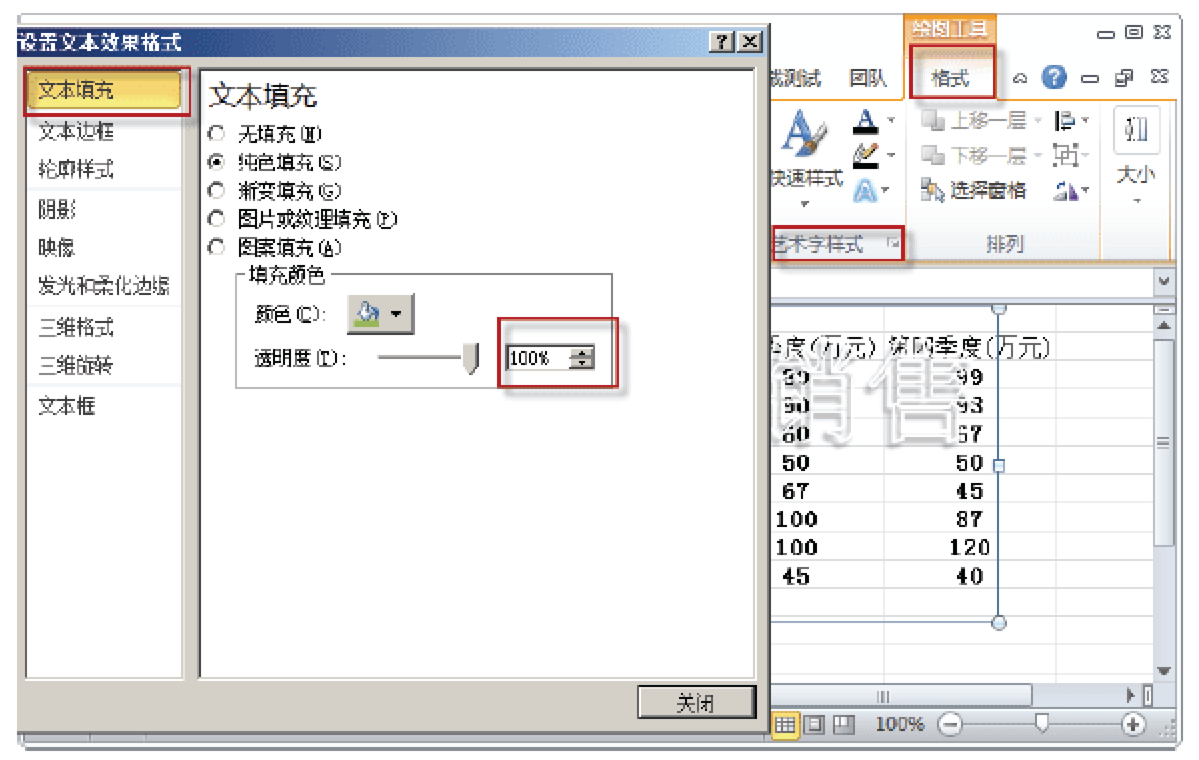


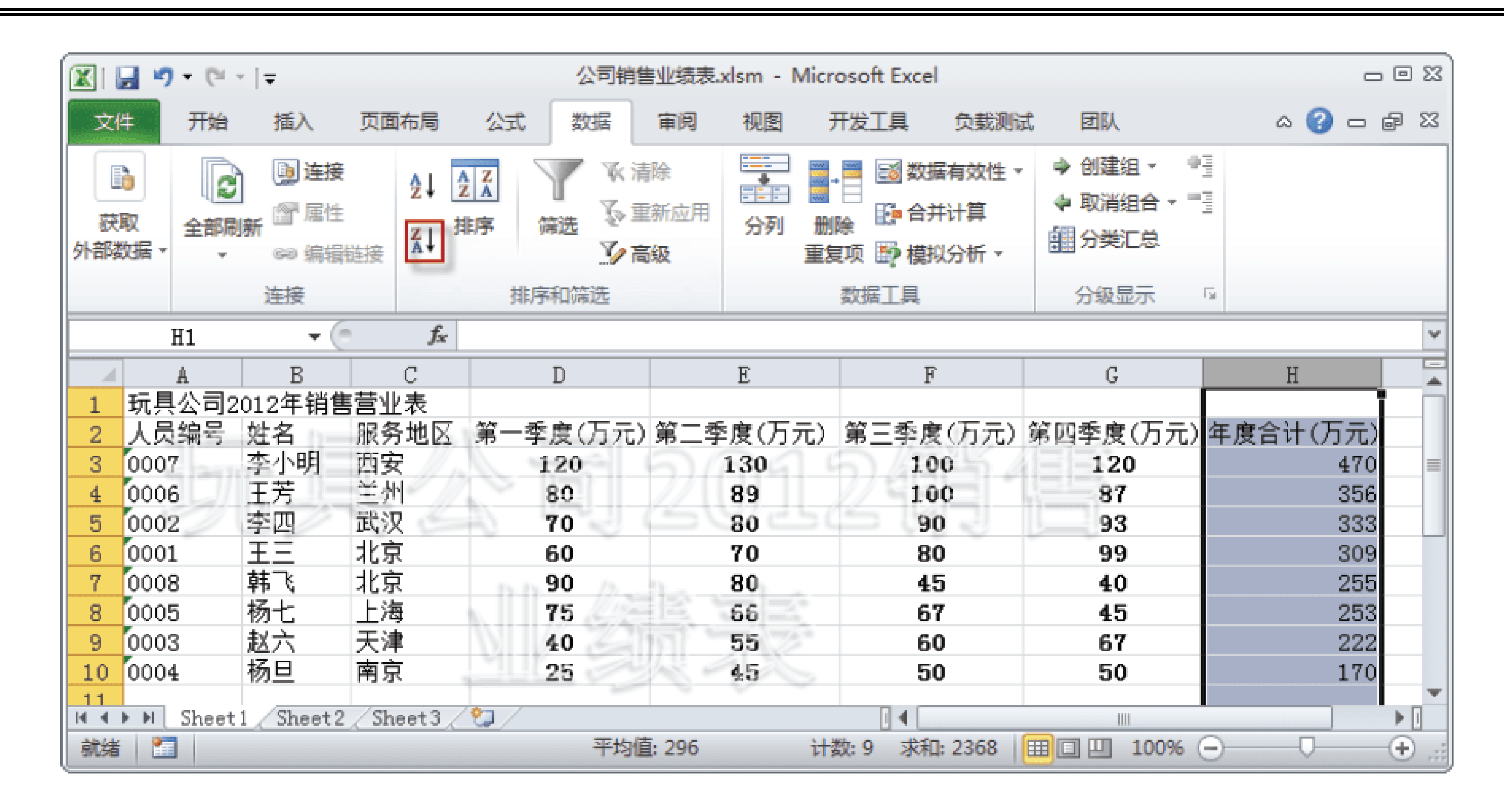
图 16.12　设置文本填充的透明度

图 16.13　设置文本边框的透明度

16.2.2　使用 Excel 2010 功能区的容器控件

功能区中的容器控件用来包含其他控件，通过容器控件的嵌套，能够在功能区中创建层次结构。常见的容器控件包括盒子控件、按钮组控件、菜单控件、组合框控件和分离按钮控件。

（1）重新显示行列标题。在工作表中选择“年度合计（万元）”列，单击“数据”选项卡的“排序和筛选”组中的“降序”按钮使工作表按照年度合计销售业绩的值降序排列，如图 16.14 所示。

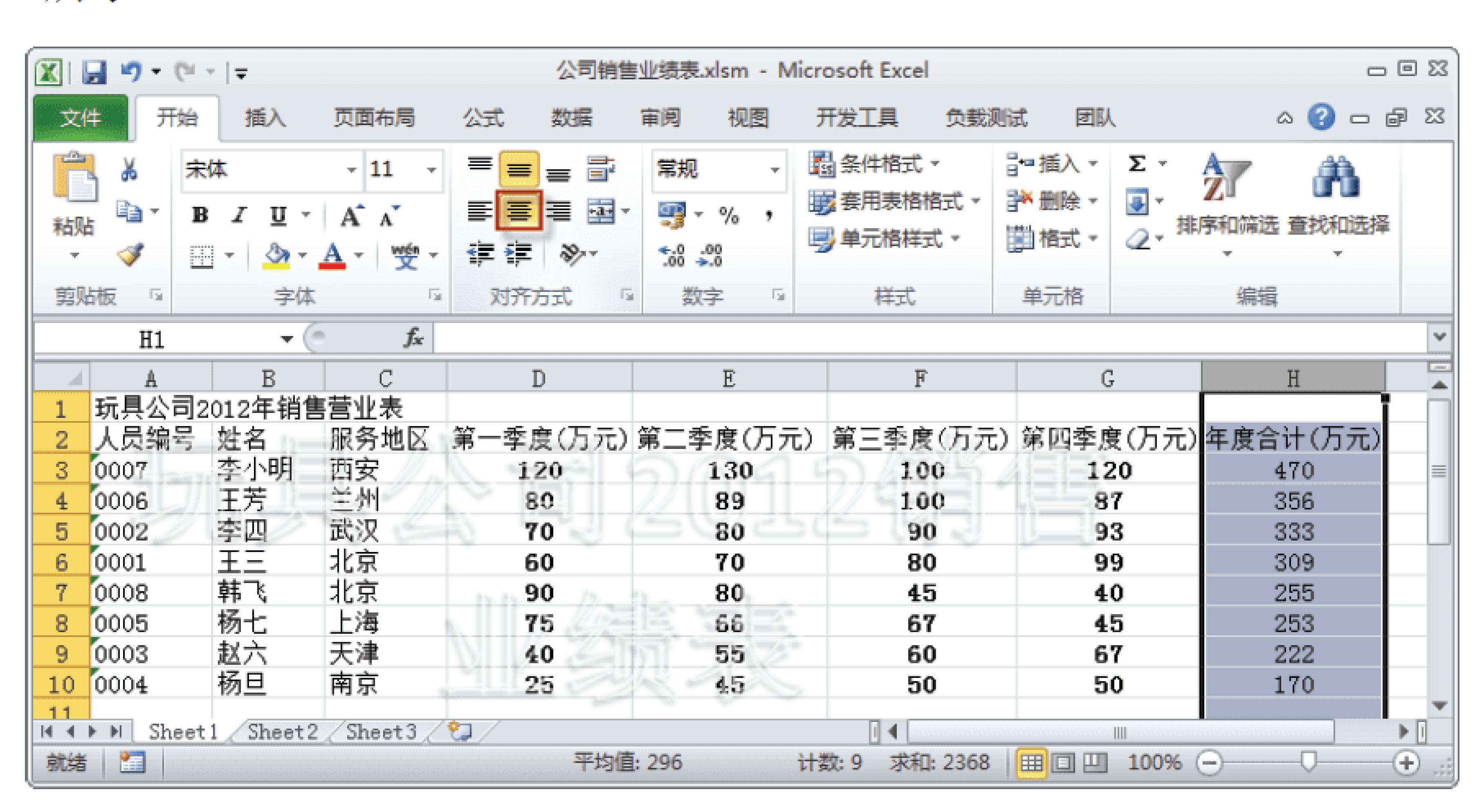


	A	B	C	D	E	F	G	H
1	玩具公司2012年销售营业表							
2	人员编号	姓名	服务地区	第一季度(万元)	第二季度(万元)	第三季度(万元)	第四季度(万元)	年度合计(万元)
3	0007	李小明	西安	120	130	100	120	470
4	0006	王芳	兰州	80	89	100	87	356
5	0002	李四	武汉	70	80	90	93	333
6	0001	王三	北京	60	70	80	99	309
7	0008	韩飞	北京	90	80	45	40	255
8	0005	杨七	上海	75	66	67	45	253
9	0003	赵六	天津	40	55	60	67	222
10	0004	杨旦	南京	25	45	50	50	170

图 16.14　按照年度合计数据执行“降序”排序

提示：在功能区中，盒子控件是不会显示出来，其主要用来控制其他控件的布局。在盒子控件中可以包含任何类型的控件，如，“数据”选项卡的“排序和筛选”组中按钮的组织就使用了盒子控件，其包含了“升序”、“降序”和“排序”功能按钮。实际上，在功能区中，像这样的盒子控件还有很多。盒子控件的 XML 标识为<box>。

（2）在“开始”选项卡中单击“居中”按钮，使年度合计数据在单元格中居中放置，如图 16.15 所示。



	A	B	C	D	E	F	G	H
1	玩具公司2012年销售营业表							
2	人员编号	姓名	服务地区	第一季度(万元)	第二季度(万元)	第三季度(万元)	第四季度(万元)	年度合计(万元)
3	0007	李小明	西安	120	130	100	120	470
4	0006	王芳	兰州	80	89	100	87	356
5	0002	李四	武汉	70	80	90	93	333
6	0001	王三	北京	60	70	80	99	309
7	0008	韩飞	北京	90	80	45	40	255
8	0005	杨七	上海	75	66	67	45	253
9	0003	赵六	天津	40	55	60	67	222
10	0004	杨旦	南京	25	45	50	50	170

图 16.15　数据居中放置

提示：与盒子控件相似，按钮组控件也是用来控制其他控件的布局，其在控件间形成边框和隔断。如，在“开始”菜单的“对齐方式”组中，使用按钮组控件实现同类功能按钮的分组放置，如图 16.15 所示。控件组控件的 XML 标识为<buttonGroup>。

（3）为了保护单元格中数据，可以将单元格锁定。选择年度合计数据所在的列，单击“开始”选项卡的“单元格”组中的“格式”按钮，在打开的菜单中选择“锁定单元格”选

项，选择的单元格将受到保护，如图 16.16 所示。

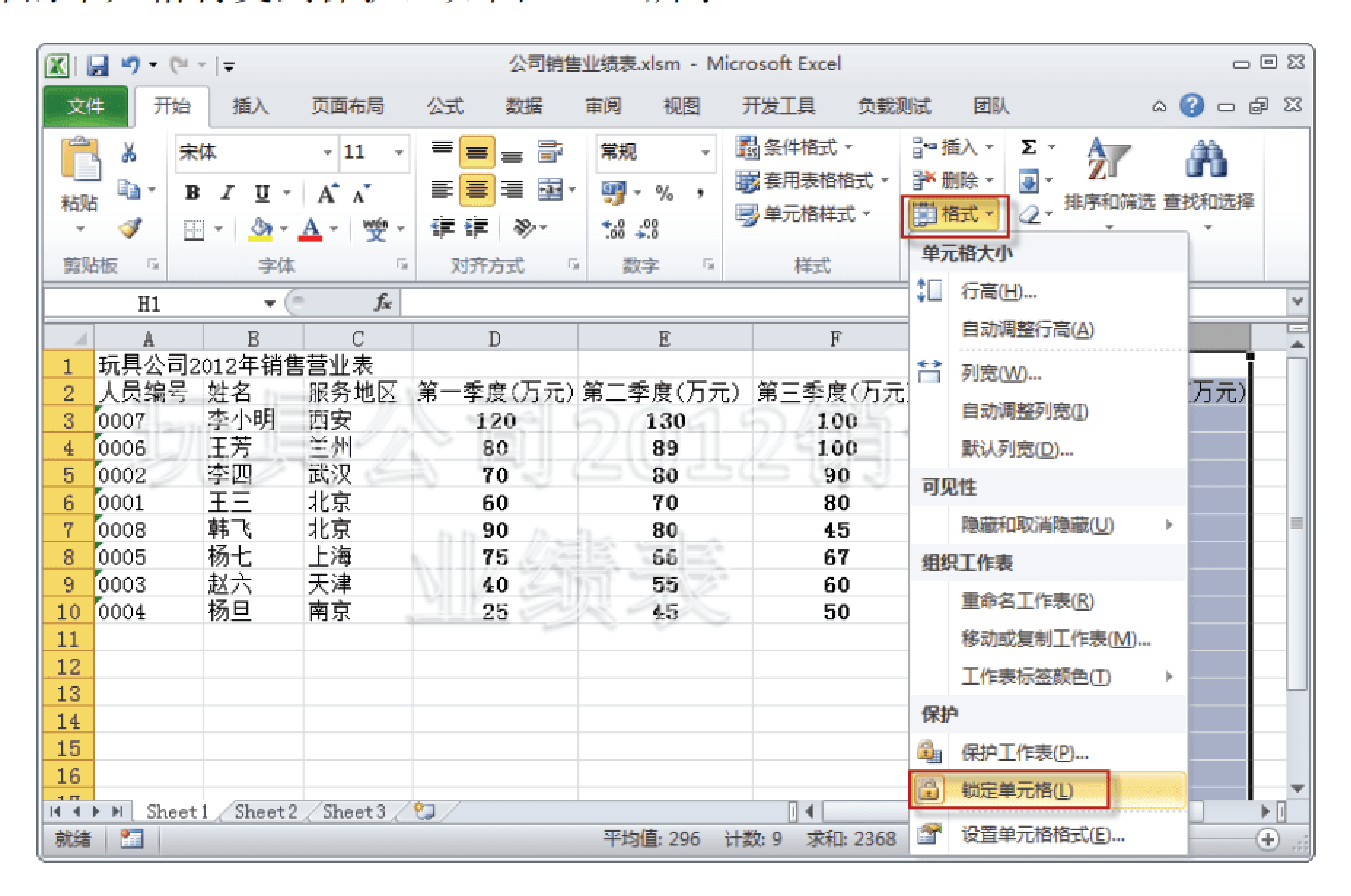


图 16.16　保护单元格

提示：这里设置单元格是否被保护，实际上使用的是菜单控件。菜单控件指的是功能区中的弹出菜单，可包含按钮或其他菜单项，能够与普通菜单一样创建层级结构。菜单控件的 XML 标识为<menu>。

（4）下面设置表中标题文字大小。在“开始”选项卡的“字体”组中单击字号组合框上的按钮，在下拉列表中选择字号，如图 16.17 所示。

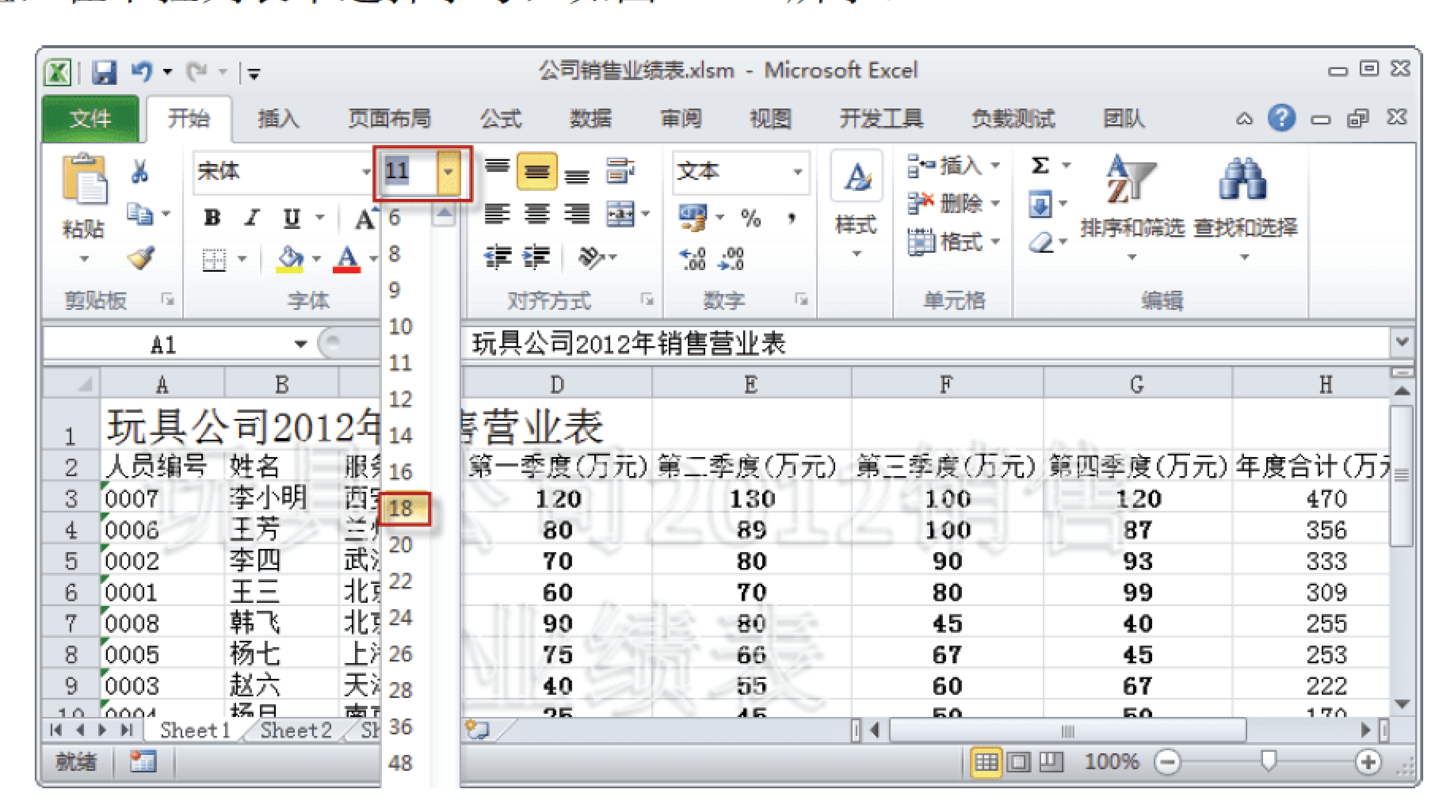


图 16.17　设置文字大小

提示：功能区中的组合框控件兼具文本框和下拉列表的功能，既可以在控件中直接输入，也可以在下拉列表中进行选择，如图 16.17 所示。组合框控件的 XML 标识为<comboBox>。

（5）选择标题文字所在的单元格，在“开始”选项卡中单击“字体”组中的边框线按钮上的下三角按钮。在打开的菜单中选择“双底框线”选项为标题添加边框线，如图 16.18 所示。

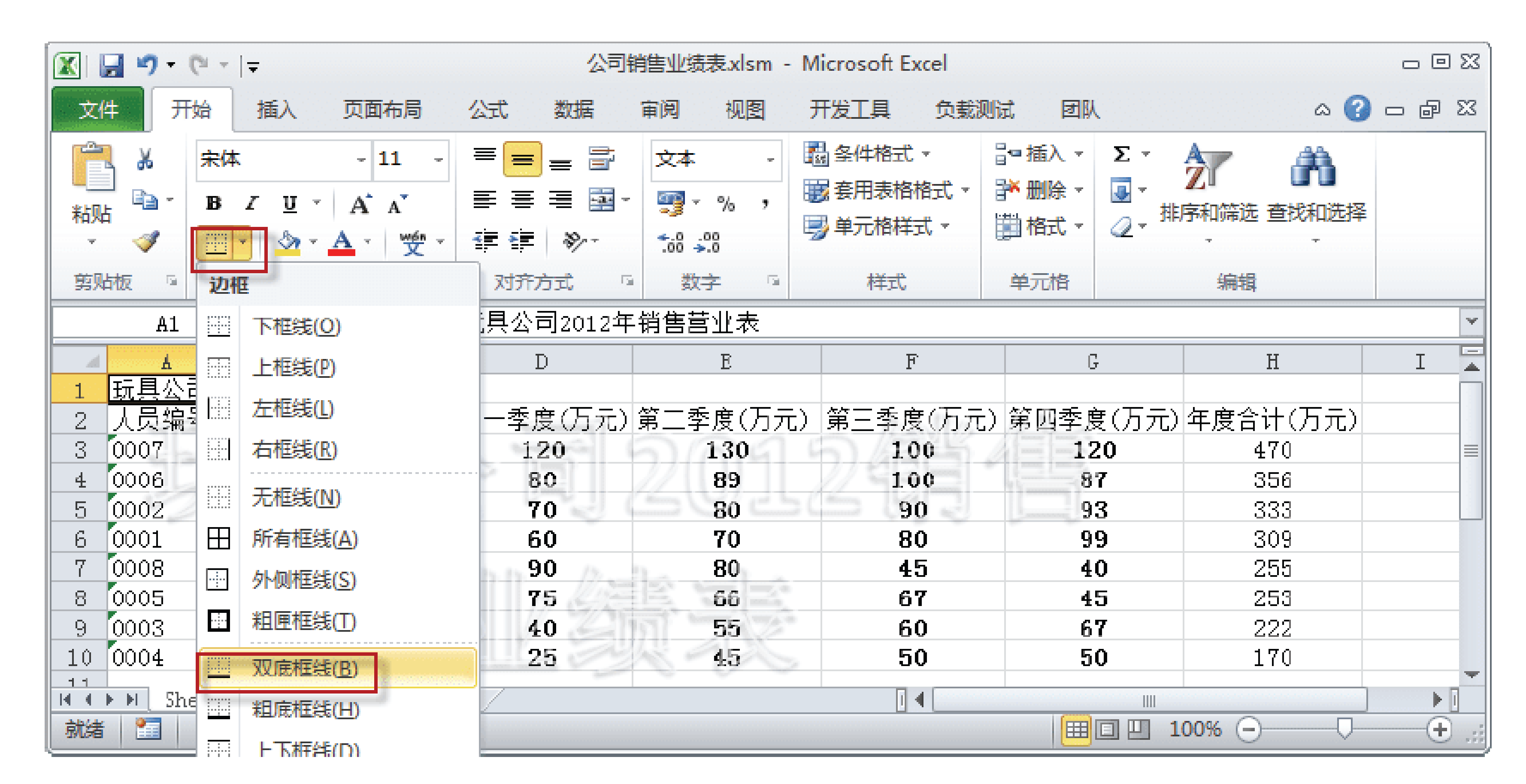


图 16.18　为标题添加边框线

🔔**提示：**这里设置边框线时使用的按钮实际上就是一个分离按钮控件。分离按钮控件是组合框（或切换按钮）与菜单的结合。单击按钮能够执行默认的操作动作，单击控件上的下三角箭头能够得到下拉列表进行选择。分离按钮控件的 XML 标识为 <splitButton>。

16.3　使用 Open XML 格式文件自定义功能区

Office 2010 提供了两种使用 XML 文件来自定义功能区的方法，一种是使用包含 XML 标识的有效 Open XML 格式文件，另一种是使用包含 XML 标记的 COM 加载项。在 XML 标记中的修改将直接表现为功能区的改变。在文档级中，使用 XML 编辑自定义功能区一般采取两个步骤，首先创建 XML 文件来定义所需要的界面样式，然后将 XML 文件插入到工作簿文件中即可。

下面通过一个实例来介绍使用 Open XML 文件来自定义功能区的方法。本实例创建一个名为“My Tools”的选项卡，在选项卡中添加“剪贴板”、“字体”、“数字”和“排列”组。同时创建一个名为 MyFunctionButton 的组，在组中添加需要的按钮。本实例的具体制作步骤如下：

（1）启动 Excel 2010，创建工作表，将工作表文件保存为“使用 Open XML 文件定义功能区.xlsm”文件后，退出 Excel。在保存该文件的文件夹中创建一个名为“customUI”的文件夹，如图 16.19 所示。

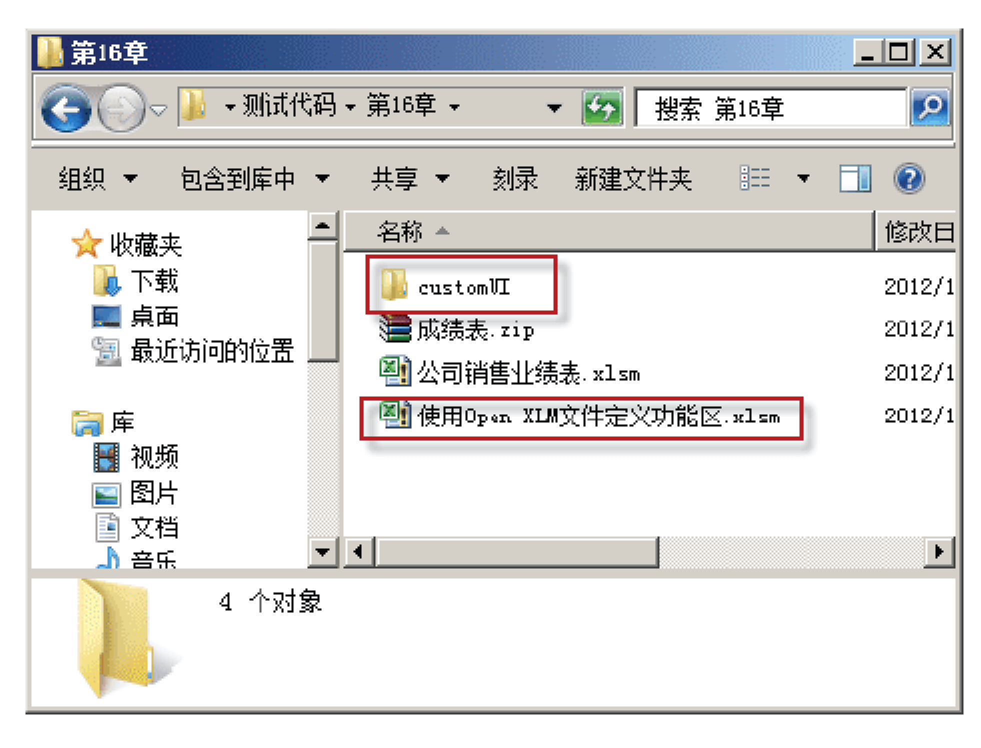


图 16.19　保存文件并创建文件夹

（2）启动记事本程序，在记事本中输入如下

XML 代码：

```
01  <customUI xmlns="http://schemas.microsoft.com/office/2009/07/
    customui">
02      <ribbon startFromScratch="false">
03          <tabs>
04              <tab id="rxtabCustom"
05                  label="我的选项卡"
06                  insertBeforeMso="TabHome">
07                  <group idMso="GroupFont">
08                  </group>
09                  <group idMso="GroupZoom">
10                  </group>
11              </tab>
12          </tabs>
13      </ribbon>
14  </customUI>
```

在第 01 行中，customUI 元素是 XML 的根容器，名称集将它识别为 RibbonX 文档。Xmlns 标识命名空间声明，程序将其声明为微软网站，其标示文件所遵循的架构版本。ribbon 元素是一个联系到可见的 Ribbon 所有变化的容器，实际上指的就是功能区。功能区包含若干选项卡，选项卡也是容器，可以包括若干个组或按钮等。每个组也是容器，可以包括若干控件和按钮。<tabs>元素是一个联系到 Ribbon 中现有或新的选项卡的所有变化的容器，也就是常说的集合，其与 VBA 中对象的规定一致。

（3）在“记事本”中选择“保存”→“另存为”命令，将文件保存为“customUI14.xml”文件，如图 16.20 所示。

（4）将步骤 1 中创建的“使用 Open XML 文件定义功能区.xlsm”文件更名为“使用 Open XML 文件定义功能区.zip”，并将 customUI 文件夹拖放到“使用 Open XML 文件定义功能区.zip”文件上，如图 16.21 所示。

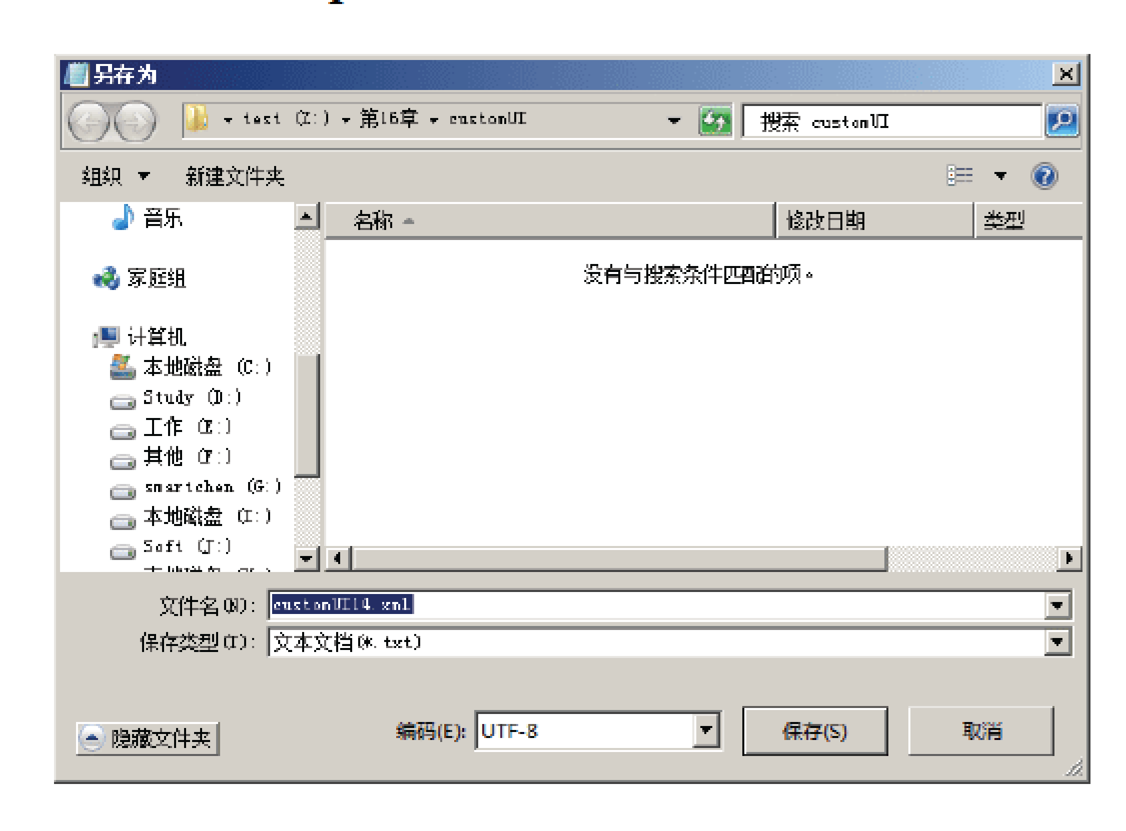


图 16.20　保存文本文件

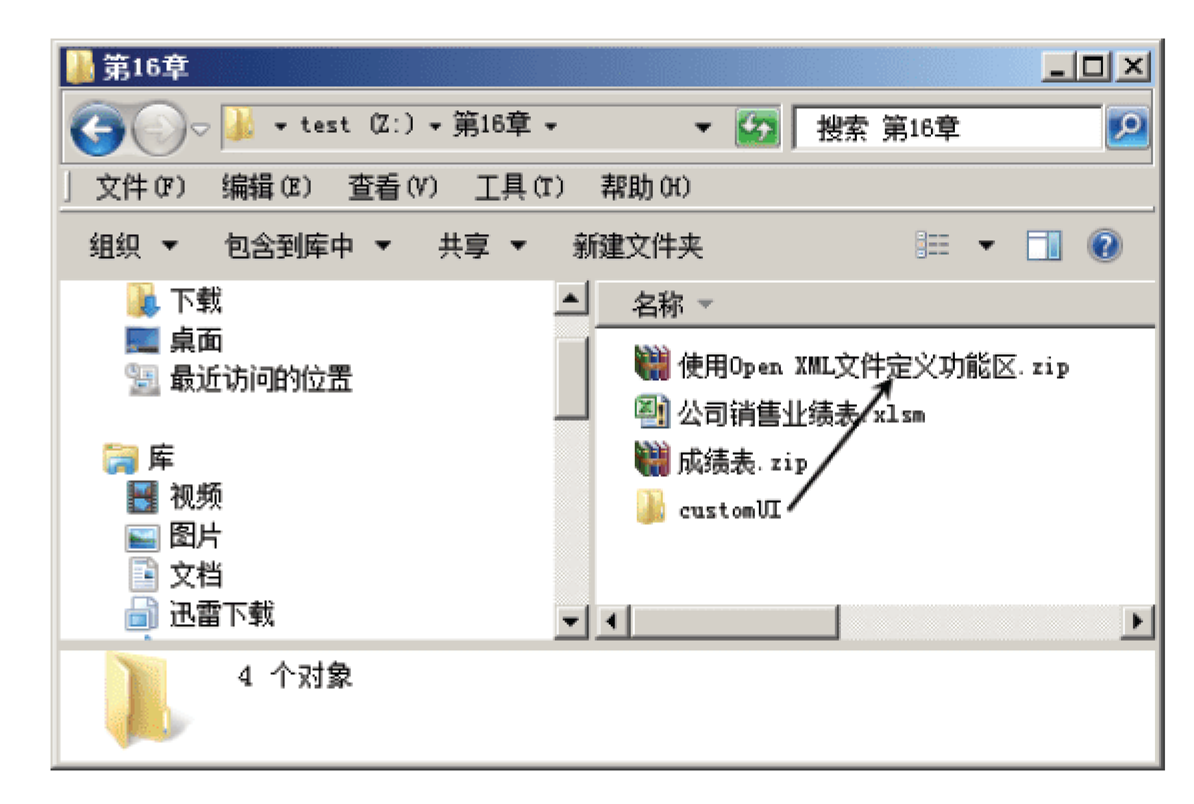


图 16.21　将文件夹拖动压缩包中

（5）双击压缩包文件打开压缩包，将其中的“_rels”文件夹拖动到当前的文件中，如图 16.22 所示。

（6）打开“_rels”文件夹，右击其中的“.rls”文件，在弹出的快捷菜单中选择“使用记事本打开”命令打开该文件，在最后一个 Relationship 标记和 Relationships 标记间添加

一行代码，此时文件中完整的 XML 代码如下所示：



图 16.22　将“_rels”文件夹拖放到当前文件夹中

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/
2006/relationships">
<Relationship Id="rId3" Type="http://schemas.openxmlformats.org/
officeDocument/2006/
relationships/extended-properties" Target="docProps/app.xml"/>
<Relationship Id="rId2" Type="http://schemas.openxmlformats.org/
package/2006/
relationships/metadata/core-properties" Target="docProps/core.xml"/>
<Relationship Id="rId1" Type="http://schemas.openxmlformats
.org/officeDocument/2006/
relationships/officeDocument" Target="xl/workbook.xml"/>
<Relationship Id="customUIRelID" Type="http://schemas.microsoft.com/
office/2006/
relationships/ui/extensibility" Target="customUI/customUI.xml"/>
</Relationships>
```

注意：上述所有代码除注释语句外，其他语句在记事本中实际上是一行，由于显示格式的原因，这里被分成了多行。

（7）完成上述代码写入后，保存文件。将“_rels”文件夹重新拖曳到“使用 Open XML 文件定义功能区.zip”文件中，替换原来的文件，如图 16.23 所示。

（8）将“使用 Open XML 文件定义功能区.zip”文件的文件名重新改为“使用 Open XML 文件定义功能区.xlsm”。双击打开该文件，在文件的功能区中添加了一个名为“我的选项卡”的选项卡，其中包括“字体”组和“显示比例”组，如图 16.24 所示。

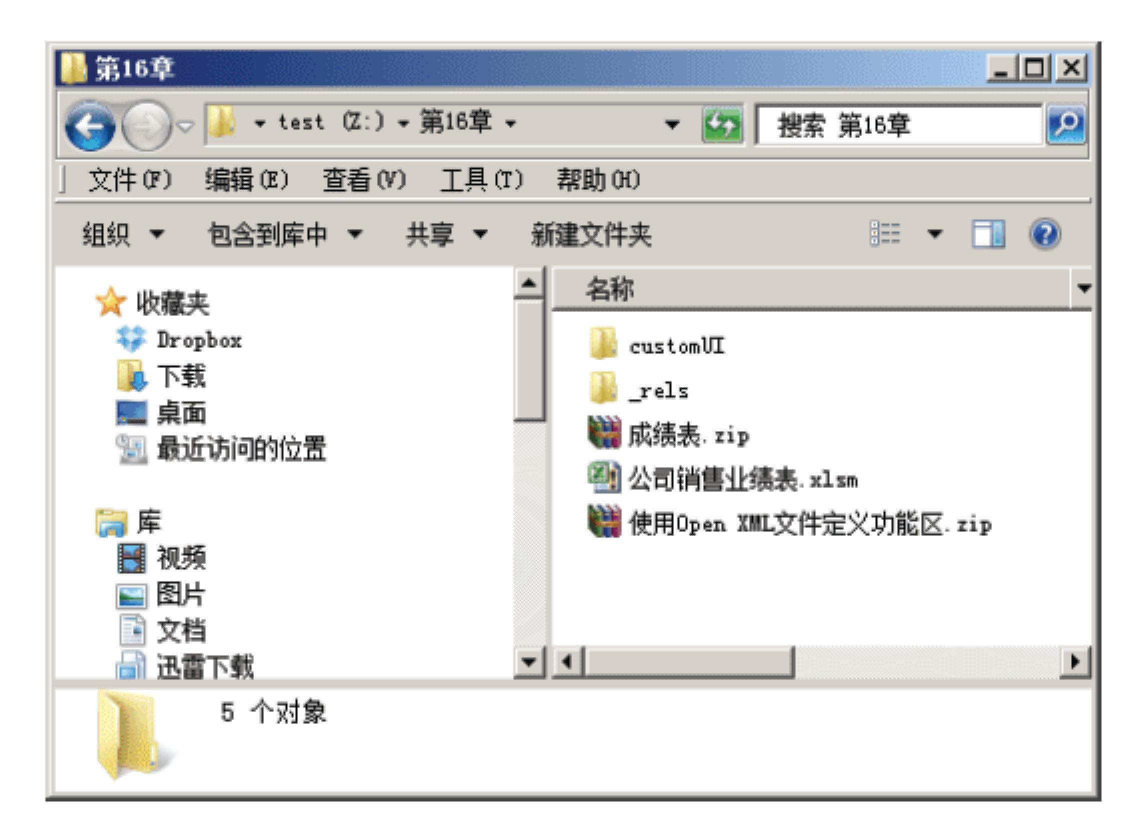


图 16.23　将“_rels”文件夹拖到压缩包中

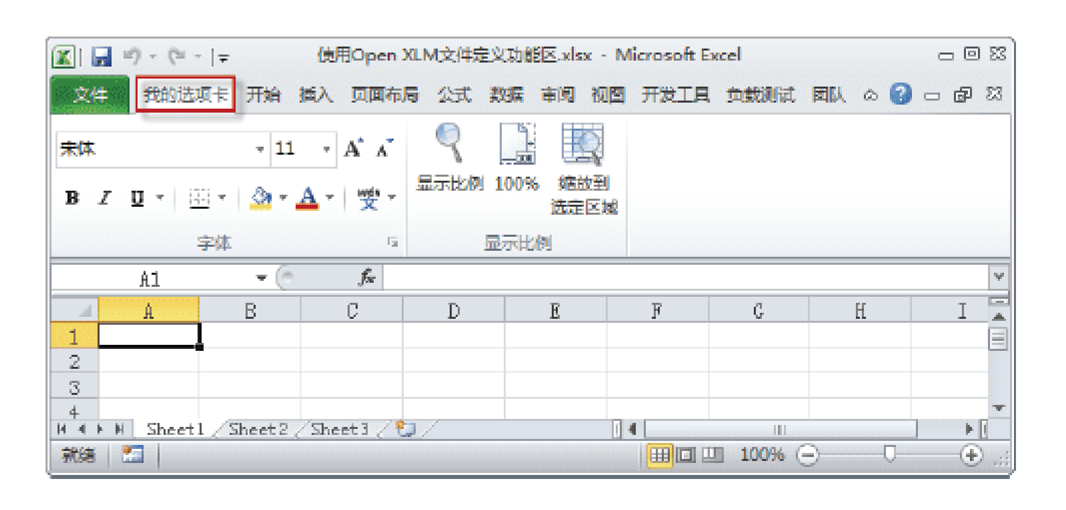


图 16.24　功能区中添加自定义选项卡

16.4　使用 UI 编辑器设计功能区

直接编辑 XML 文件自定义的功能区，操作过程较为繁琐，只要某个操作步骤出错，就会导致功能区的自定义失败。为了能够快速高效地完成功能区的定义，Microsoft 为 Office 2010 提供了一个小巧的 UI 编辑器。使用 UI 编辑器，只需要在编辑器中输入 XML 代码，保存文档后即可实现对功能区的修改。

下面通过一个实例来介绍 UI 编辑器的使用方法。本实例定制 Excel 2010 功能区，在“开始”选项卡中添加一个自定义按钮，程序运行时单击该按钮将能够调用工作表中已经创建好的宏。

（1）启动 Excel 2010，创建工作表。打开 Visual Basic 编辑器，双击 ThisWorkbook。在代码窗口中输入程序代码以实现对工作表中非标题单元格的全部选择。具体程序代码如下所示：

```
Sub selectrange(ByVal control As IRibbonControl)
    Application.Intersect(ActiveSheet.UsedRange,
    ActiveSheet.UsedRange.Offset(1, 0)).Select     '选择单元格
    MsgBox "除标题外，工作表中共有单元格" & Selection.Count & "个。"
                                                   '给出单元格个数
End Sub
```

警告： 创建宏时，必须声明宏参数类型，否则将无法从功能区中调用，如这里的过程声明中的 ByVal control As IRibbonControl 语句。

（2）将工作簿保存为“使用 UI 编辑器设计功能区.xlsm”文件，退出 Excel 2010。打开“UI 编辑器”，选择 File→Open 命令，打开“Open ODXML Document”对话框。在对话框中选择需要自定义功能区的 Excel 文档，如图 16.25 所示。

提示： 在使用 UI 编辑器时，必须首先安装该编辑器，该编辑器可以在微软网站下载，其下载页面为：http://openxmldeveloper.org/blog/b/openxmldeveloper/archive/2010/08/10/23248.aspx。在安装前，系统必须首先安装了.NetFramework，否则编辑器将无法安装，系统会提示安装.NetFramework。安装完成后，在桌面和 Windows 的“开始”→“程序”菜单中均会添加其快捷方式，使用这些快捷方式即可启动 UI 编辑器。

（3）在 UI 编辑器的编辑窗口中输入 XML 代码，如图 16.26 所示。

这里，自定义功能区的 XML 代码如下所示：

```
<---声明命名空间-->
<customUI xmlns="http://schemas.microsoft.com/office/2006/
01/customui">
<---标示功能区容器-->
  <ribbon>
    <---标示选项卡容器-->
```

```
        <tabs>
            <---创建自定义选项卡-->
            <tab id="CustomTab" label="My Own Tab">
                <---创建一个组-->
                <group id="Group1" label="My Own Tools">
                    <---创建一个按钮-->
                    <button id="Button" label="Select Range Area"
                    size="large" 07
onAction="ThisWorkbook.selectrange" />
                <---group 结束标示-->
                </group >
            <---Tab 结束标志-->
            </tab>
            <---Tabs 结束标志-->
        </tabs>
    <--ribbon 结束标志-->
    </ribbon>
  <--customUI 结束标志-->
  </customUI>
```

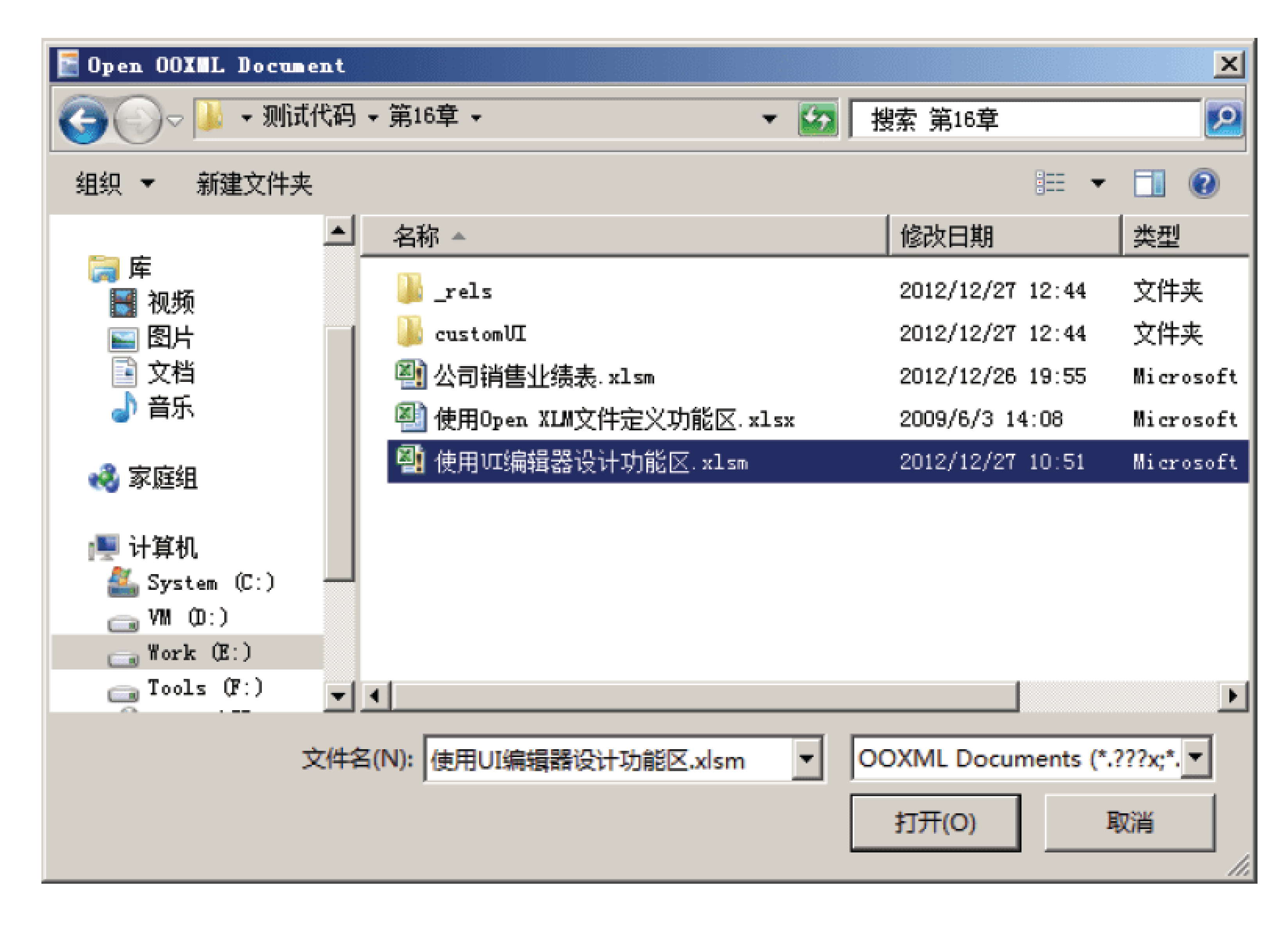


图 16.25　打开文件

```
使用UI编辑器设计功能区.xlsm - Custom UI Editor for Microsoft Office
File  Edit  Insert
使用UI编辑器设计功能
  customUI14.xml

<customUI
xmlns="http://schemas.microsoft.com/office/2009/07/customui">
  <ribbon>
    <tabs>
      <tab id="CustomTab" label="My Own Tab">
        <group id="Group1" label="My Own Tools">
          <button id="Button" label="Select Range Area"
size="large" onAction="ThisWorkbook.selectrange"/>
        </group >
      </tab>
    </tabs>
  </ribbon>
</customUI>

E:\Amateur\Excel VBA从入门到精通\测试代码\第16章\使用UI编辑器设计功能区.xlsm    Ln 7 Col 23
```

图 16.26　在 UI 编辑器中输入程序代码

提示：在这段代码中，第 08 行代码用于创建一个 My Own Tab 的选项卡。第 10 行在 My Own Tab 选项卡中生成一个名为 My Own Tools 的组。第 12 行在组中创建一个名为“Select Range Area”的按钮，其用于启动当前工作簿中的 selectrange 过程，按钮使用大按钮样式，按钮显示的名称为 Select Range Area。

（4）单击工具栏中的 Save 按钮，将输入的内容保存到 Excel 文档中。单击工具栏中的 Insert Icons 按钮，打开 Insert Custom Icons 对话框。在对话框中选择图片作为按钮图标，如图 16.27 所示。单击“打开”按钮，此时，插入的图片出现到 UI 编辑器的树形列表中，如图 16.28 所示。



图 16.27　插入图片

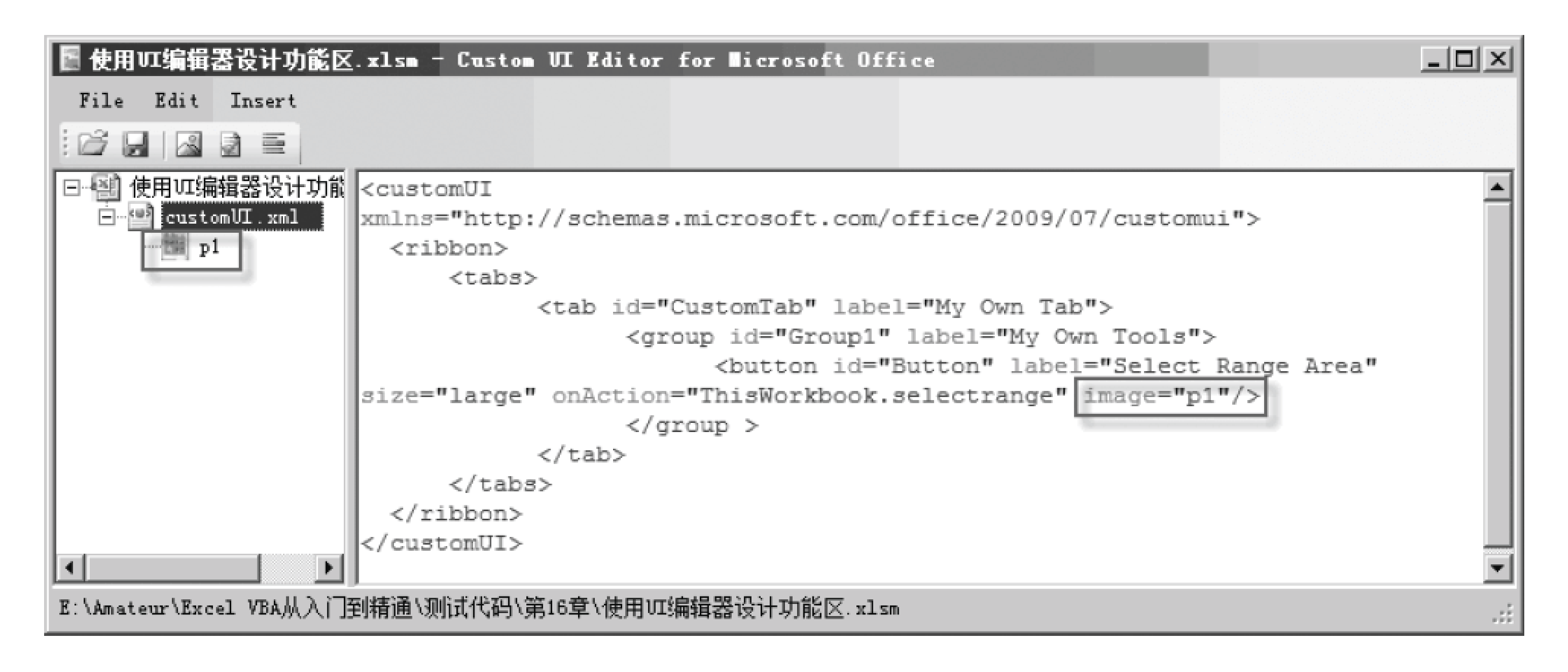


图 16.28　图片放置到 UI 编辑器树形目录

警告：在为按钮指定图片时，文档必须已经保存，否则图片将无法载入 UI 编辑器。

（5）在编辑区中修改程序代码，在代码的第 06～07 行的按钮设置语句中添加“image="p1"”，将图片指定给按钮，该行完整的 XML 代码如下：

```
01  <--创建按钮-->
```

```
<button id="Button" label="Select Range Area"
 size="large" onAction="This.Workbook.selectrange"  image="p1" />
```

（6）再次单击工具栏中的 Save 按钮保存文档。关闭 UI 编辑器，启动 Excel 2010 打开“使用 UI 编辑器设计功能区.xlsm”文件，此时在功能区中能够看到添加的选项卡和功能按钮。单击该按钮将调用宏，调用宏的效果如图 16.29 所示。

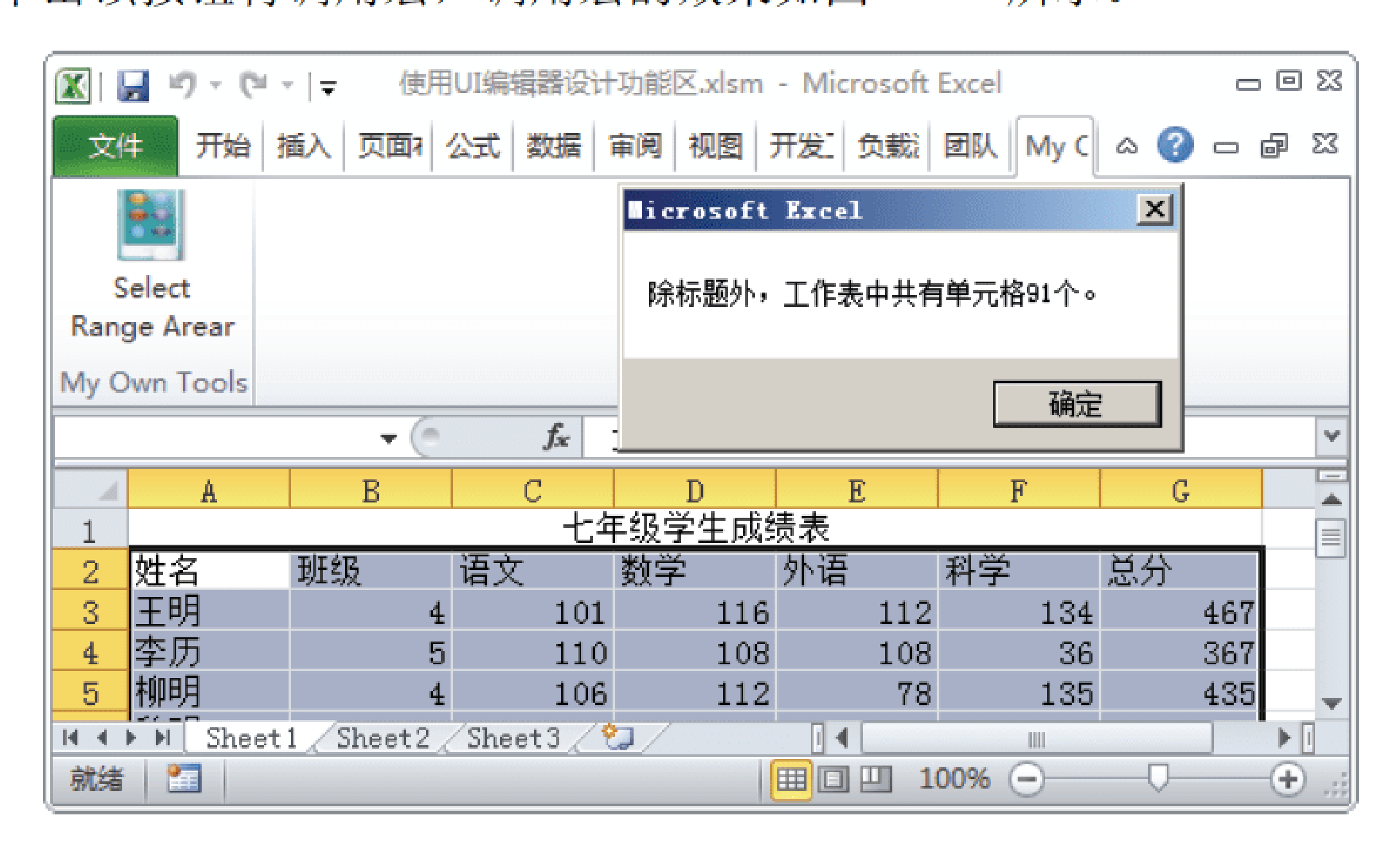


图 16.29　使用功能区自定义按钮调用宏的效果

提示： UI 编辑器是不支持中文的，在编辑器中输入的中文显示为乱码。如需要制作中文界面，可以先使用 UI 编辑器定制界面，然后使用 16.3 节的方法将 customUI.xml 文件中相应内容改为中文，将文件保存为 Unicode 格式即可。

16.5　小　　结

本章介绍了 Excel 2010 文档与 XML 文件的关系，介绍了 Excel 2010 功能区的构成特点。同时，通过实例制作介绍了直接编辑 XML 文件自定义功能区和使用 UI 编辑器自定义功能区的方法。

Excel 2010 采用了全新的 Open XML 文件格式，这种文件格式具有极强的开放性，通过对 XML 标记的修改就能完成功能区的自定义。读者如果需要设计出更为出色的功能区，则需要进一步学习 XML 语法格式。

16.6　本 章 习 题

1．下面哪项不属于功能区的容器控件？(　　)

A．盒子控件　　B．按钮组控件　　C．组合框控件　　D．按钮控件

2．下面哪项是下拉库列表的 XML 标识符？（　　）

A．<checkBox>　B．<toggleButton>　C．<separator>　D．<gallery>

3．下面语句能够实现什么功能？（　　）

```
<box id="SortBox">
    <control idMso="SortAscendingExcel" ShowLabel="False">
    <control idMso="SortDscendingExcel" ShowLabel="False">
    <control idMso="SortDialog" ShowLabel="False">
</box>
```

A．将功能区中的 3 个排序按钮水平放置
B．将功能区中的 3 个排序按钮垂直放置
C．使功能区中的 3 个排序按钮不可见
D．取消功能区中 3 个排序按钮的标签显示

4．下面语句能够实现什么功能？　（　　）

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/
01/customui">
    <ribbon>
        <tabs>
            <tab id="MyTab" label="TabOne">
                <group id="Group1" label="Delete"
                    <button id="Button" label="DeleteMe " size="large"
                    onAction="DeleteMe" />
                </group >
            </tab>
        </tabs>
    </ribbon>
</customUI>
```

A．在功能区的“加载项”选项卡中添加一个名为 DeleteMe 的按钮
B．在“快速反应工具栏”中添加一个名为 DeleteMe 的按钮
C．在功能区中添加一个名为 TabOne 的选项卡，选项卡中添加一个名为 DeleteMe 的按钮
D．在文件选项卡菜单中添加一个名为 TabOne 的菜单项，其下级菜单中添加一个名为 DeleteMe 的按钮

5．使用 UI 编辑器自定义功能区。要求在“开始”选项卡中添加一个名为 My Group 的组，组中包括 4 个按钮，同时包括一个名为 My Menu 的下拉选项菜单，菜单中包含 6 个选项。

【提示】代码中首先创建一个组，组中添加按钮，为按钮指定标签、宏启动项和显示的图标。完成组中按钮添加后，添加一个<menu>控件，然后再向<menu>控件中添加按钮。

第 17 章　控 制 图 表

Excel 2010 不仅能处理大量的数据，而且还能通过图表从不同的角度展示数据，在 Excel VBA 中使用图表对象可以设置图表的显示内容和显示方式，同时还可以控制图表的绘图区、坐标轴和数据系统，图表对象具有丰富的属性、方法和事件。本章所要学习的内容和目标如下所述：

- 掌握 Excel 2010 图表对象的引用方法；
- 掌握 Excel 2010 图表对象的属性和事件，能够在程序中添加图表对象；
- 掌握操作 Excel 2010 图表对象的图表区、绘图区、坐标轴、数据系列的方法。

17.1　引用 Excel 图表对象

Excel 中的图表分为两类，一类是嵌入式图表，另一类是图表工作表。嵌入式图表浮在工作表的上面，位于工作表的绘图层中。而图表工作表则占据整个工作表，也就是说在工作表中只包含工作表对象，这种图表工作表常在需要打印整页图表时使用。

在 VBA 中，Chart 对象表示工作簿中的图表，这个图表既可以是嵌入式图表，也可以是单独的图表工作表。多个 Chart 对象就组成了 Charts 对象集合，Charts 集合包含了工作簿中每个工作表中的 Chart 对象，不同的工作簿将会有不同的 Charts 集合。这样，可以通过 Charts 对象集来直接引用某个图表，其语法如下所示：

```
Charts(Index)
```

这里，Index 是图表工作表的索引号或名称，其表示图表工作表在工作簿标签栏上的位置。所有的图表工作表都包括在索引数内，即使是隐藏的工作表也包括在内。例如，下面两个语句分别表示图表工作簿的第一个和最后一个图表：

```
Charts(1)
Charts(Charts.Count)
```

Chart 对象同时也是 Sheets 对象集合的成员，此集合包含了工作簿中所有的工作表。通过索引号，也可以从 Sheets 对象集中指定图表工作表，其语法如下所示：

```
Sheets(Index)
```

如果在 Excel 2010 工作表中选择了一个图表工作表，或者使用 Activate 方法激活了工作表，则图表工作表也可以使用 ActiveChart 属性来引用。比如，在“立即”窗口中显示图表工作表的名称，可以使用下面的语句：

```
Print.Debug ActiveChart.Name
```

在 Excel VBA 中，ChartObject 代表工作表中的嵌入图表。ChartObject 对象是 Chart 对象的容器，可以使用 ChartObject 对象的属性和方法来对嵌入图表进行控制。ChartObject 对象是 ChartObjects 对象集的成员，该对象集表示单一工作表中的所有嵌入图表。使用索引号或对象名称能够引用 ChartObjects 对象集中的 ChartObject 对象，其语法格式如下所示：

```
ChartObjects(Index)
```

17.2　创建 Excel 图表对象

对象的属性、方法和事件是对象编程的三要素。使用图表对象的 Add 方法能够创建对象，使用图表对象的属性能够对图表对象进行设置。本节将介绍图表对象常见的属性、方法和事件，并说明使用它们创建和设置图表的方法。

17.2.1　使用 Excel 图表对象的常见属性

使用图表对象的属性能够对工作表中图表的外观样式进行设置。在 Excel 2010 中，图表的类型一共有 73 种，可以使用图表对象的 ChartType 属性来设置，如，其值为 xl3DArea 表示三维面积图，其值为 xlBarOfPie 表示复合饼图，其值为 xlBuble 表示气泡图。

注意：Excel 2010 的 73 种图表类型所对应的参数，可在 VBA 帮助文档的“枚举”对象的“XlChartType 枚举”中查到。

在 VBA 中，ChartTitle 属性将返回一个 ChartTitle 对象，使用该对象属性和方法可以对工作表标题进行操作，如，使用 Text 属性能够设置图表的标题文字。

注意：只有当 Chart 对象的 HasTitle 属性设置为 True 时，Text 属性设置的标题才能显示。

【范例 17-1】 在工作表中添加一个图表，使用“组合框”表单控件控制图表类型，使用“单选按钮”表单控件来控制是否显示标题。使用“命令按钮”表单控件来控制样式的更改。范例代码如下所示：

```
Sub 图表外观简单设置()
    Select Case Sheet1.Range("N8")                     '判断组合框选项
    Case 1                                             '选择第一个选项
        Sheet1.ChartObjects(1).Chart.ChartType = xl3DPie'设置为三维饼图
    Case 2                                             '选择第二个选项
        Sheet1.ChartObjects(1).Chart.ChartType = xlBubble3DEffect
                                                       '设置为三维气泡图
    Case 3                                             '选择第三个选项
        Sheet1.ChartObjects(1).Chart.ChartType = xlConeCol
                                                   '设置为三维柱形圆锥图
    Case 4                                         '选择第四个选项
        Sheet1.ChartObjects(1).Chart.ChartType = xlCylinderCol
                                                   '设置为三维柱形圆柱图
    Case 5                                         '选择第五个选项
        Sheet1.ChartObjects(1).Chart.ChartType = xlDoughnut
```

```
                                                      '设置为圆环图
13        Case 6                                      '选择第六个选项
14            Sheet1.ChartObjects(1).Chart.ChartType = xlPieExploded
                                                      '设置为分离型饼图
15        End Select
16        Select Case Sheet1.Range("p1")              '判断单选按钮状态
17        Case 1                                      '选择显示标题
18            Sheet1.ChartObjects(1).Chart.HasTitle = True   '允许标题显示
19            Sheet1.ChartObjects(1).Chart.ChartTitle.Text = Range("a1")
                                                          '指定标题内容
20        Case 2                                          '选择不显示标题
21            Sheet1.ChartObjects(1).Chart.HasTitle = False  '不允许标题显示
22        End Select
23    End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口，输入上述代码。在工作表中使用“组合框”控件选择图表类型，单击“单选按钮”控件选择是否显示标题。设置完成后单击“更改外观”按钮更改图表外观。本范例的运行效果如图 17.1 所示。

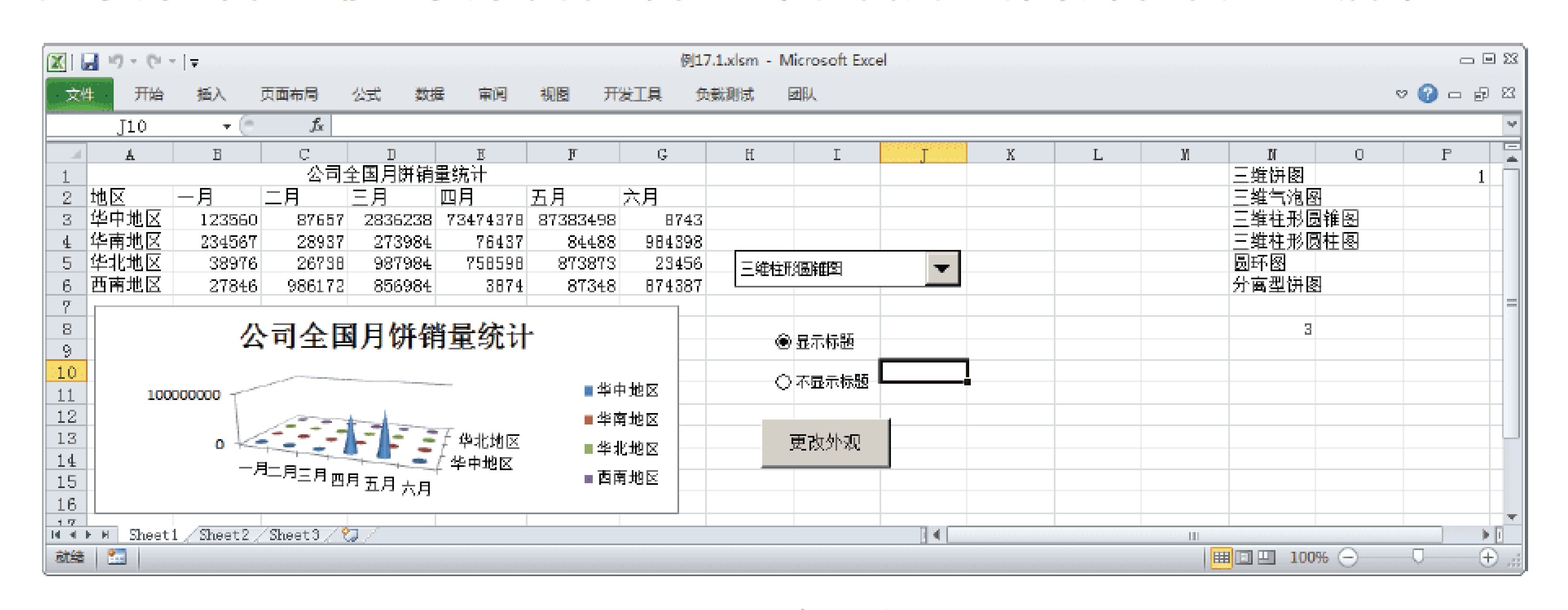


图 17.1　程序允许效果

【代码解析】本程序演示常见图表类型的设置方法。代码第 02 行使用 Sheet1.Range ("N8") 语句指定与“组合框”控件关联链接单元格，该单元格中的值对应控件的选项。第 02~15 行使用 Select Case 结构来判断“组合框”控件的选项，根据选项值来设置图表的样式。同样地，第 16～22 行代码，根据 Sheet1.Range("p1")单元格的值来判断“单选按钮”控件的选择状态，根据选择来确定图表标题是否显示。

注意：ChartTile 对象表示图表的标题，通过操作 ChartTitle 对象来实现对工作表中图表标题的设置。除了在范例中使用 Text 属性设置标题文字外，还可以使用 Left 属性和 Top 属性设置标题的位置。使用 HorizontalAlignment 属性和 VerticalAlignment 设置标题在水平方向和垂直方向上的对齐方式。使用 Delete 方法来删除标题。

17.2.2　添加 Excel 图表对象

在工作表中使用图表，首先需要添加图表。Excel 2010 中添加图表可以是嵌入式图表，也可以是插入式图表，针对创建的图表类型的不同，VBA 提供了不同的对象方法。

当需要在活动工作表指定位置上创建一个嵌入式图表时，可以使用 Shapes 对象集的 AddChart 方法。该方法有 5 个参数，其中，Left、Top、Width 和 Height 参数的值可以指定

图表的位置和大小，Type 参数的值可以指定创建图表的类型。AddChart 方法的语法结构如下：

```
表达式.AddChart(Type,Left,Top,Width,Height)
```

当需要在指定的工作簿中创建插入式工作图表时，可以使用 Chart 对象的 Add 方法。该方法使用 4 个可选参数，其中，Before 和 After 参数用于指定插入的图表与当前工作表的位置关系，是位于当前工作表之前还是之后。Add 方法的 Count 参数用于指定添加工作表的数目，Type 参数用于指定图表类型。Add 方法的语法结构如下所示：

```
表达式.Add(Before, After, Count, Type)
```

ChartObjects 对象集合同样拥有一个 Add 方法，该方法能够在指定的图表工作表、对话框编辑表或工作表的 ChartObjects 对象集合中添加图表。Add 方法有 4 个参数，其中的 Left 和 Top 参数用于指定图表相对于工作表 A1 单元格左上角或图表左上角的坐标，Width 和 Heigth 参数用于指定对象的大小。该方法的语法结构如下所示：

```
表达式.Add(Left, Top, Width, Height)
```

【范例 17-2】编程实现在工作表中依据框选的数据来创建三维圆柱图。范例代码如下：

```
Sub 创建图表()
    Dim rng As Range                                    '声明变量
    set rng = Application.Selection                     '将选择区域赋予变量
    Charts.Add                                          '添加图表
    With ActiveChart
        .ChartType = xlCylinderCol                      '图表类型为三维圆柱图
        .SetSourceData Source:=rng, PlotBy:=xlColumns  '设置图表数据源
        .Location Where:=xlLocationAsObject, Name:="Sheet1"
                                                        '设置图表的位置
    End With
    ActiveChart.ChartArea.Select                        '选取当前图表
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。在工作表中添加一个“单选按钮”控件，将创建的模块指定给这个按钮。首先在工作表中选择数据区域，然后单击“创建图表”按钮，就可以创建以选择区域数据为数据源的图表。本范例的运行效果如图 17.2 所示。

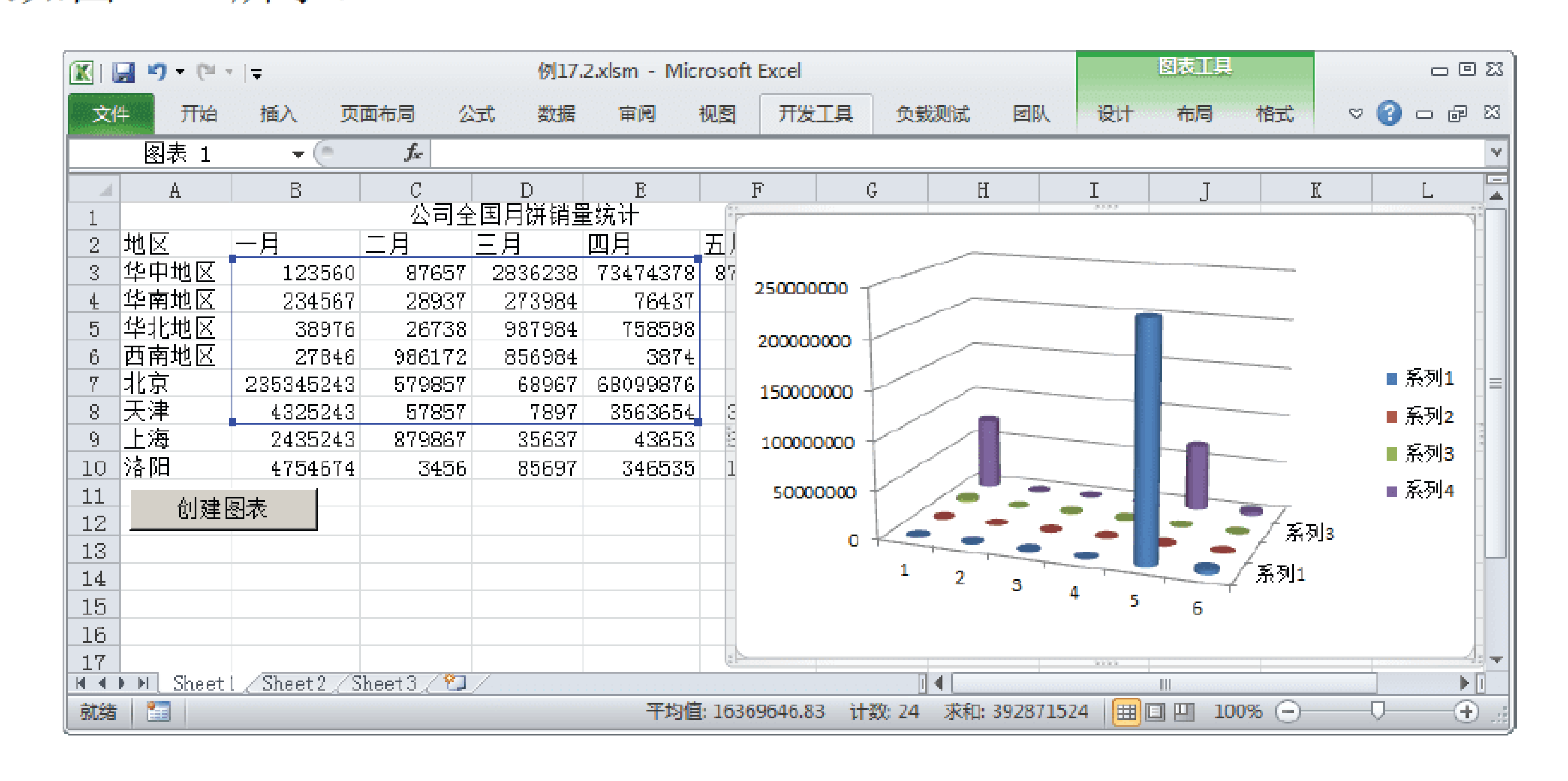


图 17.2　程序运行效果

【代码解析】在本范例程序中，使用 Chart 对象的 Add 方法来创建一个图表对象。程序的第 06 行通过 ChartType 属性设置工作表对象的图表类型，第 07 行使用 SetSource 方法指定图表的数据源。第 08 行代码使用 Location 方法将图表移到“Sheet1”工作表中，第 10 行代码使用 Select 方法选择工作表中的活动图表。

17.2.3　使用 Excel 图表对象的事件

图表作为插入工作表的对象，同样有自己的事件，如，当激活图表时，Activate 事件会触发；当图表接收新的数据时，Calculate 事件会触发。编写图表对象的事件响应程序，比一般对象的事件响应程序的编写要复杂，图表位置不同，编写事件响应程序的方式也不相同。

在创建图表时，如果图表占据整个工作表，则该图表就是一个图表工作表。图表工作表的事件响应程序的编写比较简单，可以直接编写代码来实现事件响应。下面通过一个具体的实例来介绍事件程序的编写方法。本实例仍使用范例 17-2 中创建的图表，当每次激活该图表时，图表自动随机更改背景颜色。

（1）范例 15-2 创建的图表是一个嵌入图表，下面首先将其转换为图表工作表。方法是在工作表上右击，在弹出的快捷菜单中选择“移动图表”命令。此时打开“移动图表”对话框，选中“新工作表”单选按钮，在其后的文本框中输入工作表名称。完成设置后单击“确定”按钮，如图 17.3 所示。此时，将可以创建一个工作表图表，如图 17.4 所示。

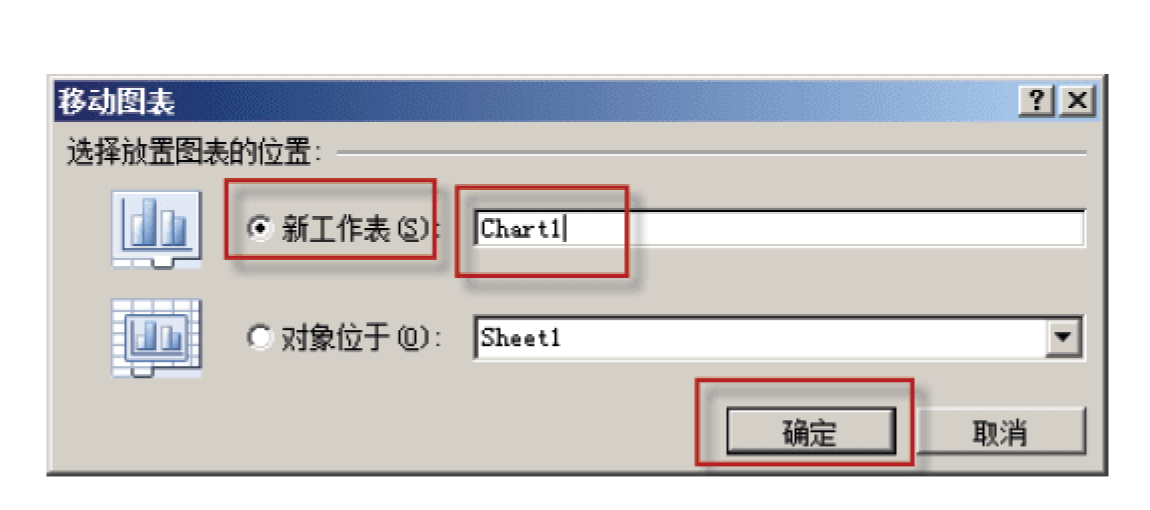


图 17.3　“移动图表”对话框

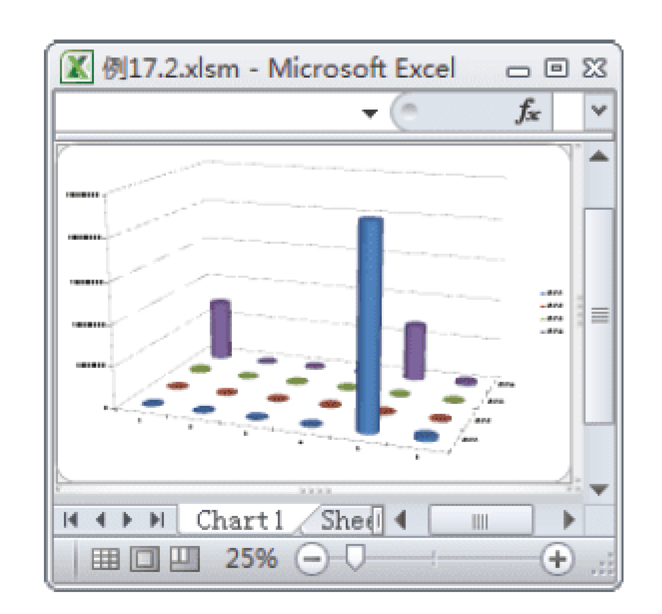


图 17.4　创建工作表图表

（2）切换到 Visual Basic 编辑器，在工程资源管理器中将会出现刚才创建的工作表图表，如图 17.5 所示。双击该图表对象，打开“代码”窗口，为该对象选择事件，如图 17.6 所示。

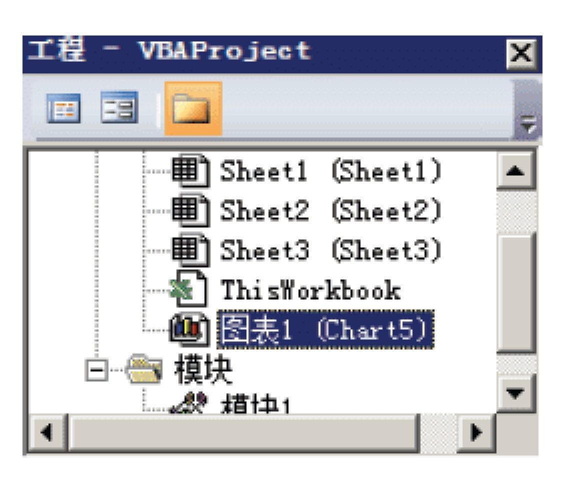


图 17.5　出现图表对象

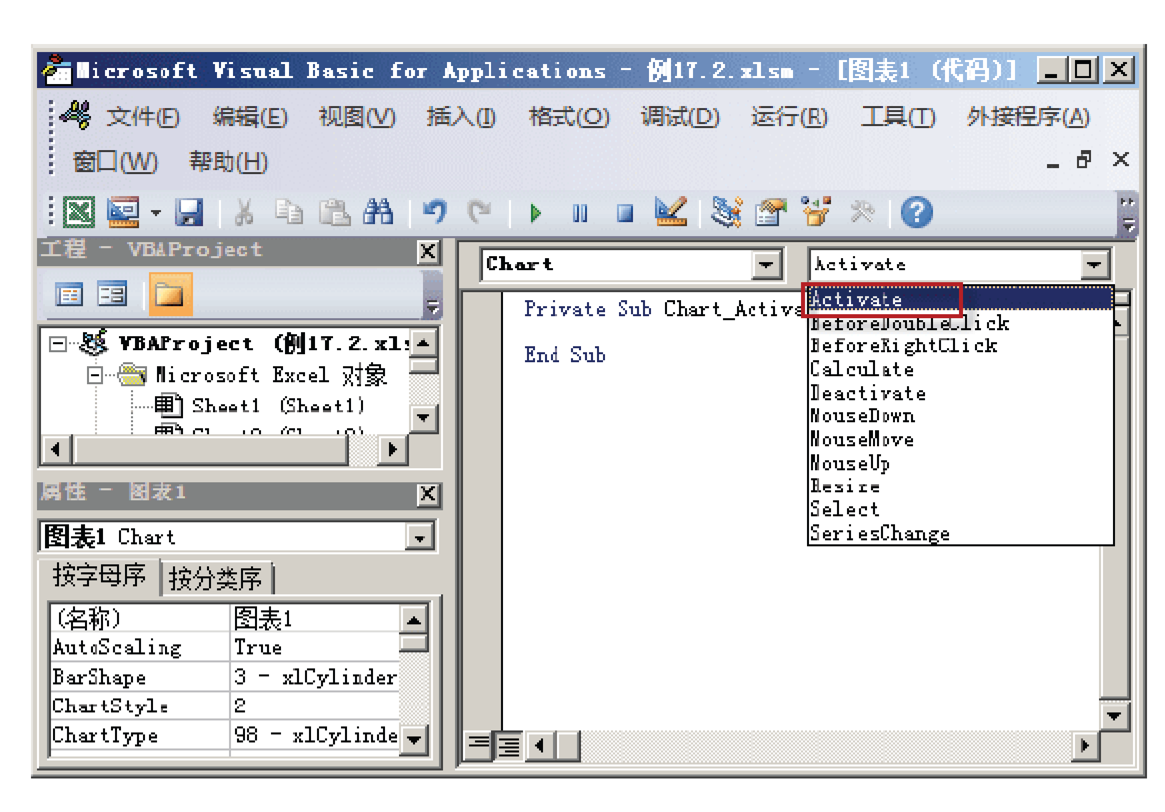


图 17.6　选择对象事件

（3）为对象添加事件代码，该代码使图表被激活后随机改变背景颜色，如图 17.7 所示。

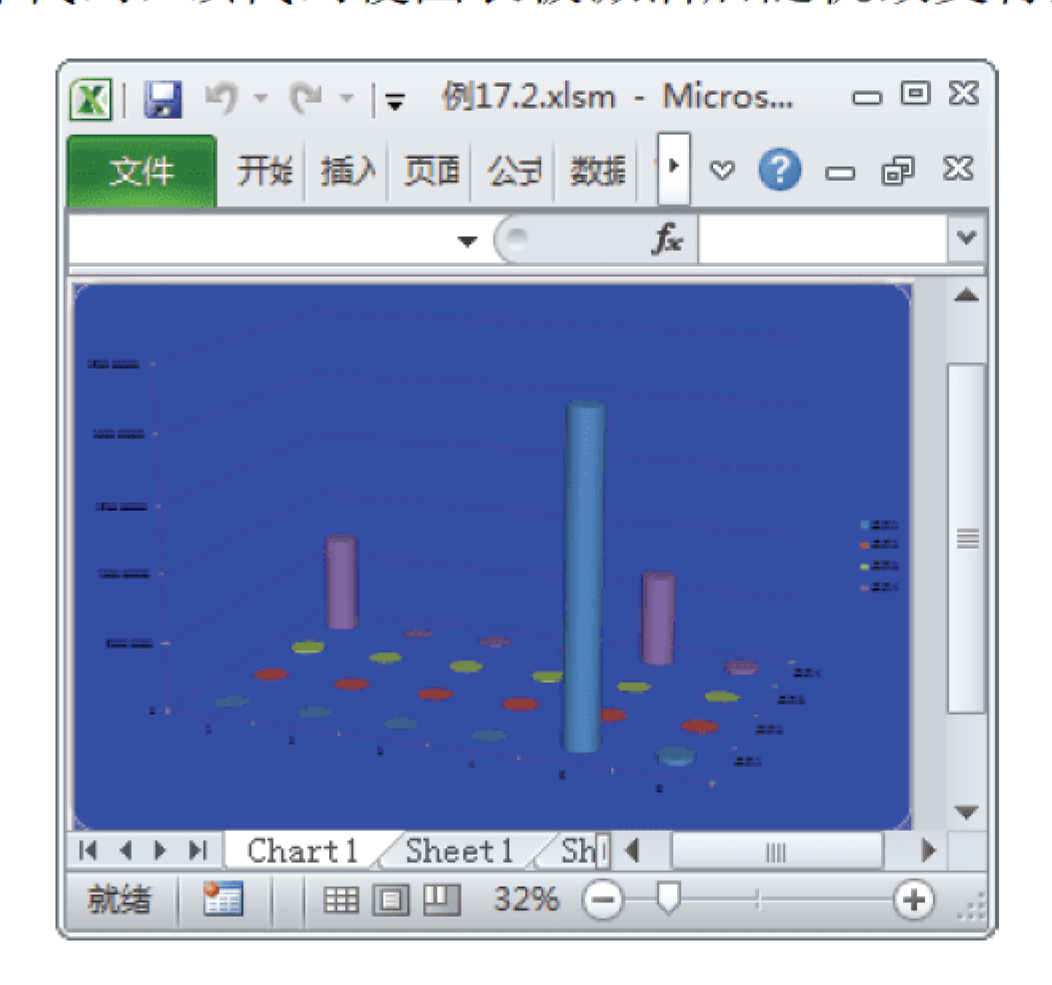


图 17.7　图表改变背景颜色后的效果

图表对象的 Activate 事件代码如下所示：

```
Private Sub Chart Activate()
    Charts(Chart1).ChartArea.Interior.ColorIndex = Rnd * 10
                                                    '随机更改背景颜色
End Sub
```

提示：代码中的 ChartArea 属性将能够返回一个 ChartArea 对象，该对象表示整个图表区。

对于嵌入式图表来说，其位于工作表中，发生的事件被工作表截获，无法直接传递给图表。这里，可以借助于类模块来解决这个问题。下面通过一个实例来介绍嵌入图表事件响应的创建方法。实例使用图表的 Select 事件，当选择图表时，图表背景颜色自动改变。下面介绍具体的操作步骤。

（1）打开工作簿，在工作簿中创建图表。启动 Visual Basic 编辑器。在工程资源管理器中右击，然后在弹出的快捷菜单中选择“插入”→“类模块”命令，插入一个类模块。在“属性”对话框中，将类模块的名称改为 myclass，如图 17.8 所示。

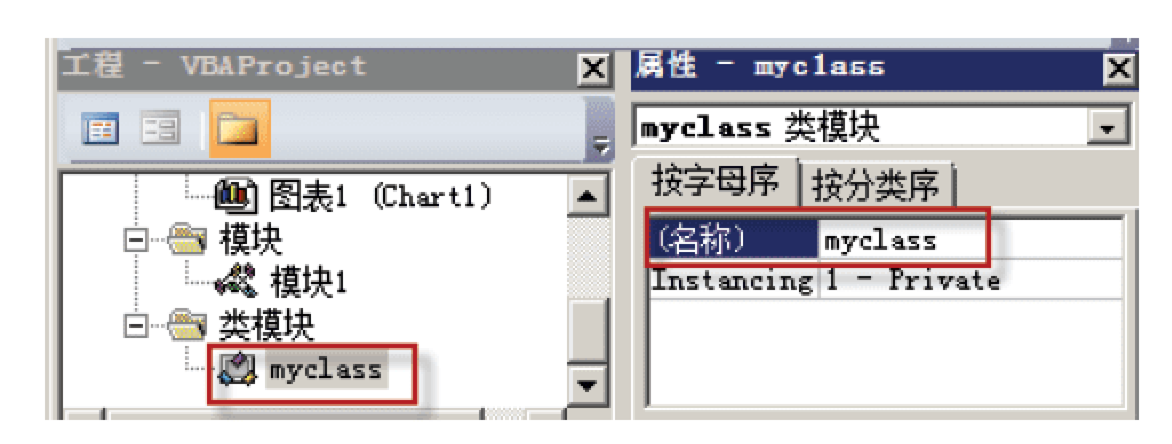


图 17.8　添加类模块并重命名

（2）在类模块的“代码”窗口中输入代码，定义一个公有变量，代码如下所示：

```
Public Withevents myclass as Chart      '声明变量
```

提示：这里，代码中的 Withevents 关键字用于说明 myclass 变量是用来响应由 Chart 对象触发事件的变量，这种声明方式只有在类模块中才是合法的。

（3）在工程资源管理器中添加一个模块，然后在模块的“代码”窗口中输入如下代码：

```
Dim cht As New myclass                                  '声明模块变量
Sub 图表事件()
    Set cht.myclass = Worksheet("sheet1").ChartObjects(1).Chart
                                                        '建立与图表的链接
End Sub
```

（4）再次回到类模块的“代码”窗口编写图表的事件代码。该段代码使用图表的 Select 事件，即当选择图表时触发事件。事件触发后，显示一个提示对话框，然后像上个例子那样更改图表背景颜色，颜色是随机设置的。详细的程序代码如下所示：

```
Private Sub myclass Select(ByVal ElementID As Long, ByVal Arg1 As Long,
 ByVal Arg2 As Long)
    MsgBox "你选择了图表，图表背景颜色将改变！"             '给出提示
    Worksheets("sheet1").ChartObjects(1).Chart.ChartArea.Interior. _
    ColorIndex = Rnd * 10                               '更改背景颜色
End Sub
```

（5）此时，选择工作簿中图表，图表的事件不会触发，这是因为并没有实现类与图表的链接。打开第 3 步创建模块的“代码”窗口，按 F5 键运行该过程建立类与图表的链接。切换到 Excel 2010 工作簿，选择图表，图表事件被触发，将显示提示对话框，如图 17.9 所示。关闭对话框后，图表背景颜色将改变，如图 17.10 所示。

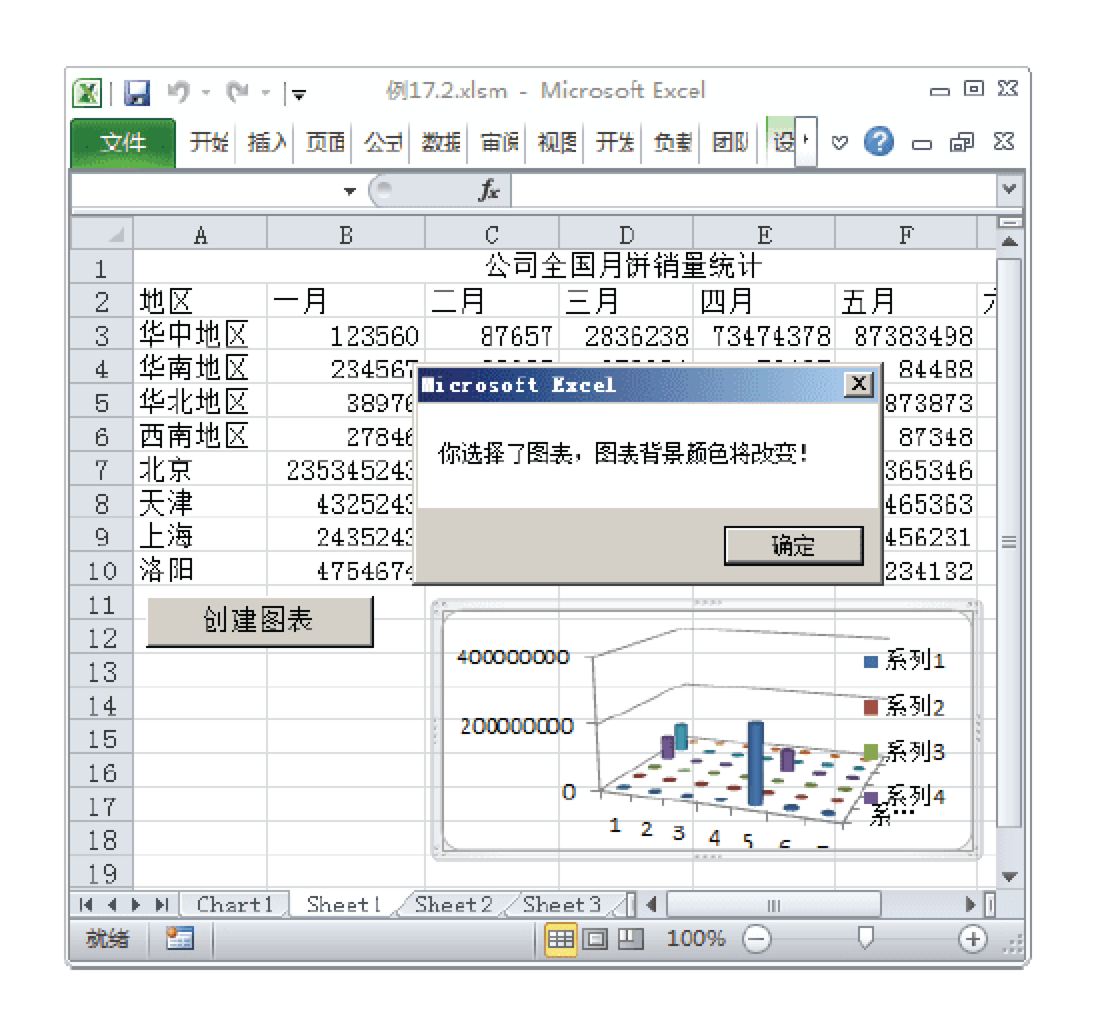




图 17.9　程序给出提示

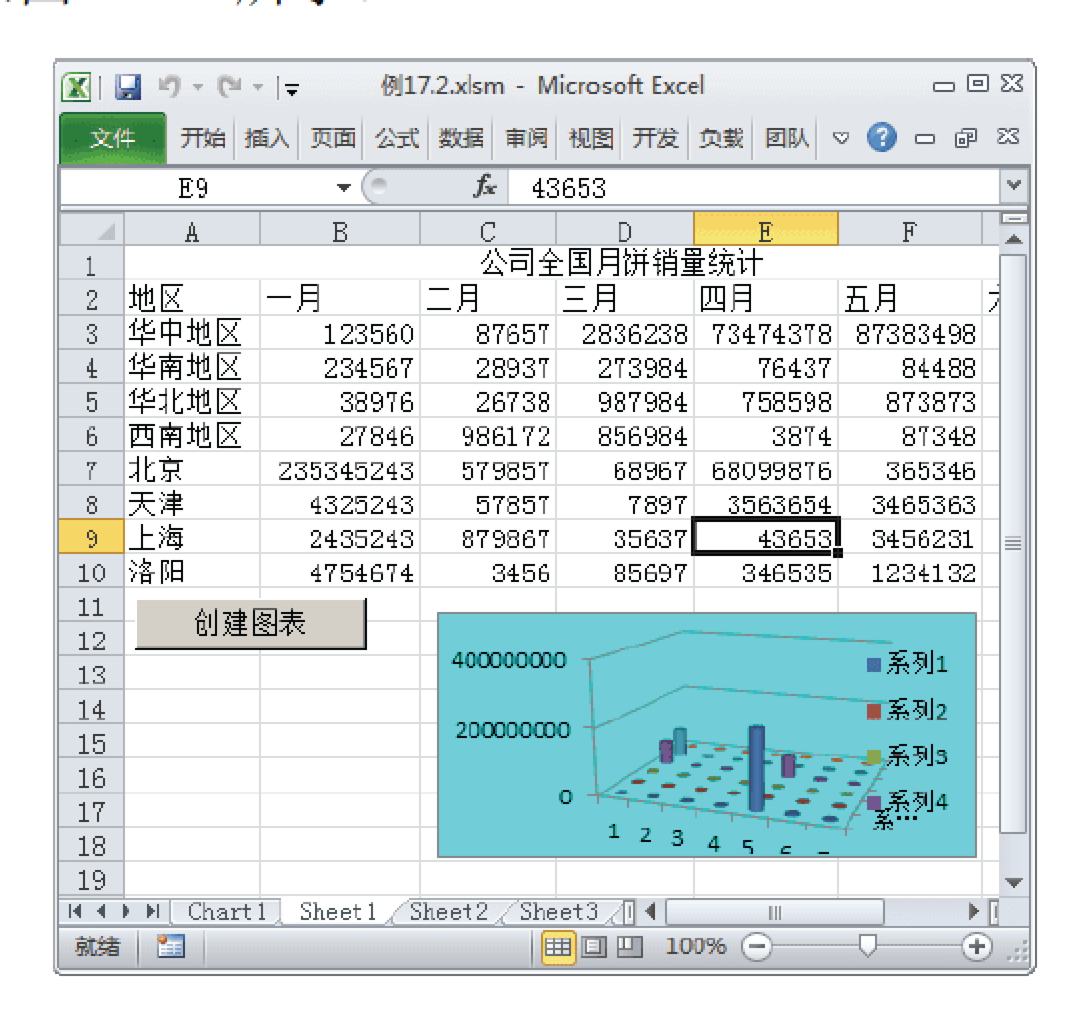


图 17.10　图表背景颜色改变

17.3　操作 Excel 图表

一个完整的图表，包括很多的构成元素，图表的操作实际上就是对这些构成元素的操作。在图表中，元素放置在图表区中，图表区包括绘图区、坐标轴、图例、数据系列、标题和数据标志等。在 VBA 中，ChartArea 对象代表图表的图表区，使用图表对象的 ChartArea 属性能够返回 ChartArea 对象。通过使用 ChartArea 对象的属性和方法就能实现对图表的进

一步操作。本节将结合一些典型的实例来介绍对图表进行操作和设置的技巧。

17.3.1　操作图表区

图表区是图表中所有其他元素的容器，图表区包括绘图区、坐标轴、图例、系列、标题和数据标志等。对图表区的操作主要是设置其格式，如边框、填充颜色、图案以及图表字体等。在 VBA 中，ChartArea 对象代表图表区，可以使用图表对象的 ChartArea 属性获得该对象。

在 VBA 中，图表区的位置是可以通过程序来设置的，这可以使用 ChartArea 对象 Left 和 Top 属性来实现。ChartArea 对象的 Left 属性代表从对象左边缘到工作表的 A 列左边缘或到图表上的图表区左边缘的距离。Top 属性代表从对象的上边缘到工作表第一行顶部或图表上的图表区顶部的距离。这两个属性均以磅为单位。

与用户窗体一样，ChartArea 对象同样具有 Height 属性和 Width 属性，通过修改这两个属性值，可以设置图表区的宽度和高度，改变图表区的大小。

【范例 17-3】 将图表区边框设置为虚线，并将图表区移到工作表的左上角，范例代码如下所示：

```
Sub 设置图表区()
    Sheet1.ChartObjects(1).Activate              '激活图表
    With ActiveChart.ChartArea
        .Border.LineStyle = xlDash               '设置图表区边框为虚线
        .Top = 0                                 '设置图表顶端位置
        .Left = 0                                '设置图表左侧位置
    End With
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行上述代码，将图表移到工作表的左上端，如图 17.11 所示。

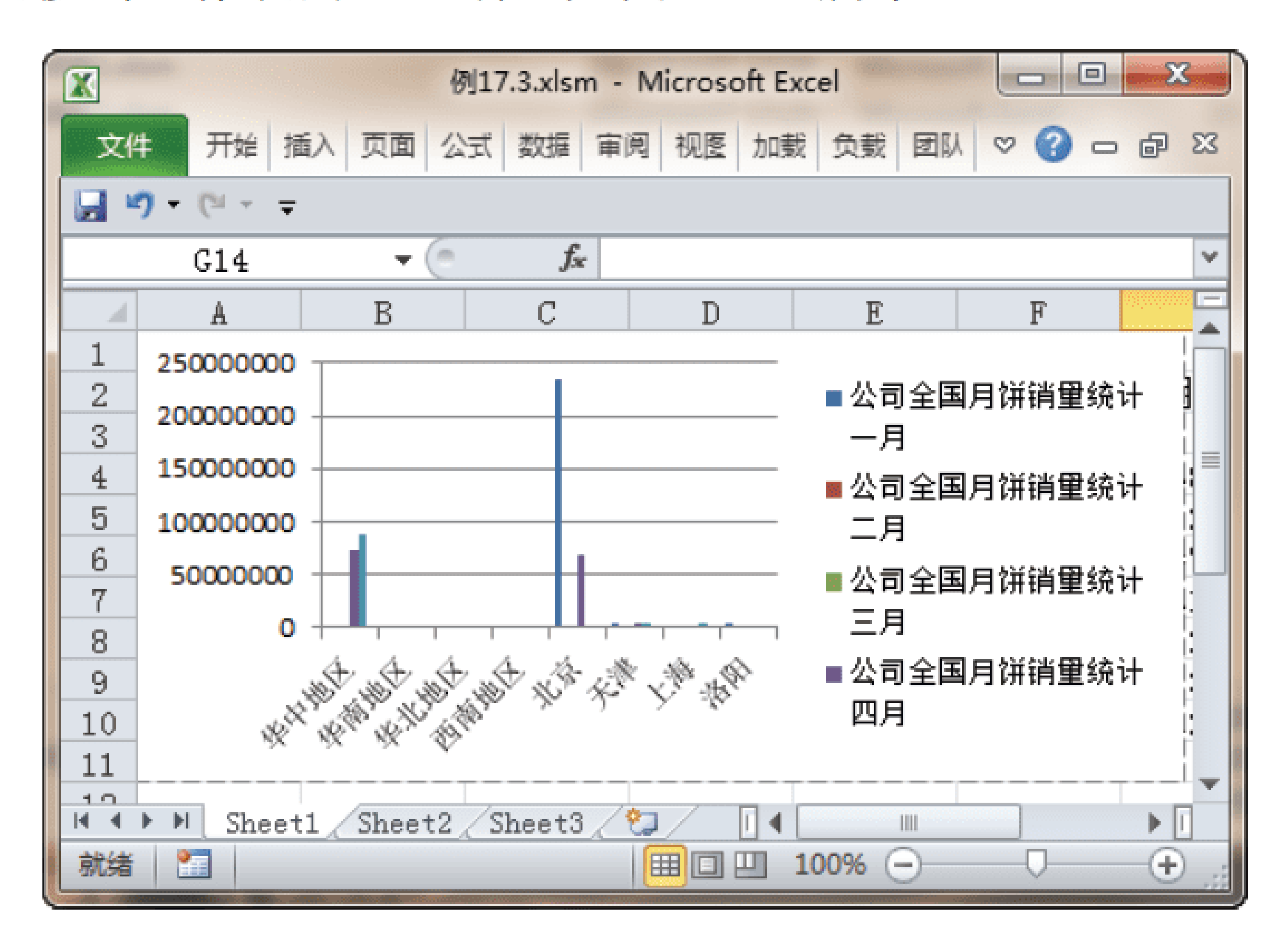


图 17.11　程序运行效果

【代码解析】本范例演示 CharArea 对象的使用。程序通过设置 ChartArea 对象的 Top 和 Left 属性来改变图表区在工作表的位置，通过设置 Border 对象的 LineStyle 属性来设置

图表区边框的样式。

17.3.2　操作绘图区

绘图区是图表中绘制图表的区域，对于二维图表来说，该区域包括数据标志、网格线和数据标签等。而三维图表的绘图区除包括上述项目外，还包括背景墙、基底、坐标轴、坐标轴标题和刻度线标签等。对绘图区的操作，可以使用 VBA 的 PlotArea 对象的属性和方法来实现。这里，简单介绍改变绘图区位置和大小的方法。

PlotArea 对象的 InsideHeight 属性和 InsideWidth 属性以磅为单位返回绘图区内部高度和宽度，改变它们的值，可以设置绘图区的高度和宽度。而 PlotArea 对象 InsideTop 属性和 InsideLeft 属性以磅为单位，返回从图表边界至绘图区内部上边界和左边界的的距离，更改它们的值将能够改变绘图区的位置。

【范例 17-4】 编程改变绘图区的大小。这里将绘图区缩小到原来的一半，并移动绘图区到图表区的中心。范例代码如下：

```
Sub 设置绘图区大小和位置()
    Sheet1.ChartObjects(1).Activate                        '激活图表
    With ActiveChart.PlotArea                              '设置绘图区
        .InsideHeight = .InsideHeight * 0.5                '设置高度
        .InsideWidth = .InsideWidth * 0.5                  '设置宽度
        .InsideTop = (ActiveChart.ChartArea.Height _
        - .InsideHeight) / 2                               '移到高度一半的位置
        .InsideLeft = (ActiveChart.ChartArea.Width _
        - .InsideWidth) / 2                                '移到宽度一半的位置
    End With
End Sub
```

【运行结果】在工作表中创建图表，如图 17.12 所示。切换到 Visual Basic 编辑器，在工程资源管理器中创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行上述代码，绘图区中的图形缩小一半，同时图形居中放置，如图 17.13 所示。

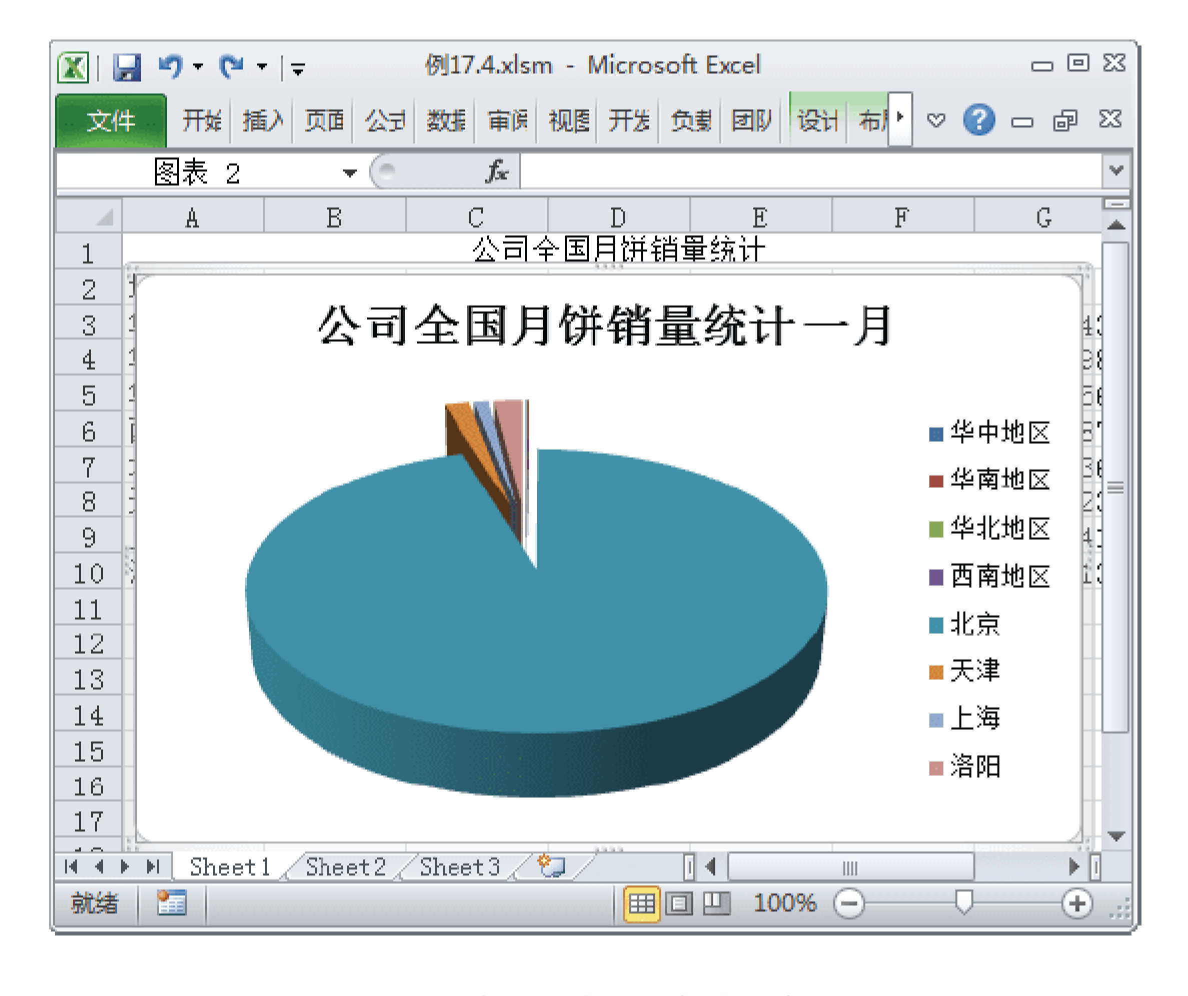


图 17.12　在工作表中创建图表

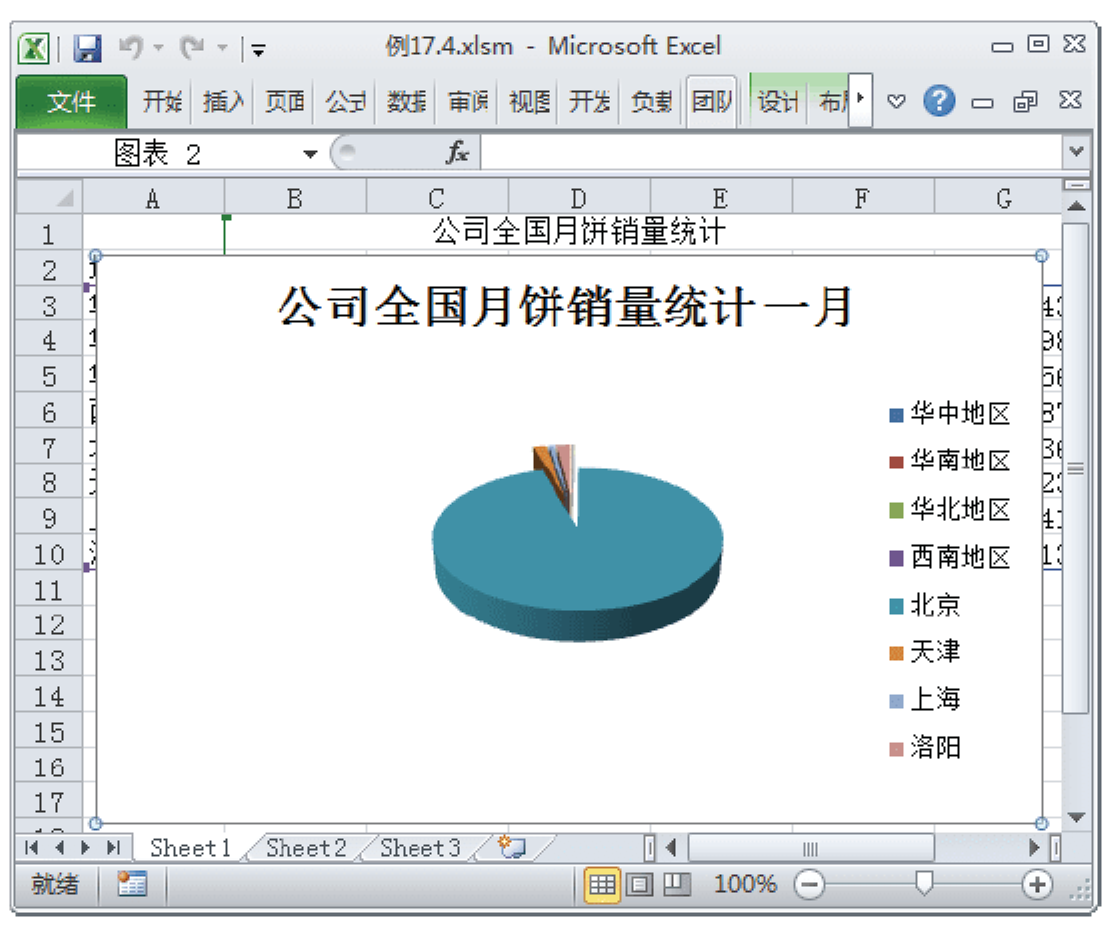


图 17.13　程序运行后的图表效果

【代码解析】本范例演示 PlotArea 对象的使用。程序使用该对象的 InsideHeight 属性和

InsideWidth 属性设置图形的大小。使用 InsideTop 属性和 InsideLeft 属性设置图形在图表区中的位置。这里要注意第 06 行和第 08 行对象居中放置的算法。

17.3.3　操作坐标轴

在一个二维图表中，图表一般拥有两个坐标轴，即 X 轴和 Y 轴，它们分别称为分类轴和数值轴。在 VBA 中，可以使用 Axis 对象来对图表的坐标轴进行设置。Axis 对象代表图表中单个的坐标轴，其为 Axes 对象集合的一个成员。如果需要获得图表中的 Axis 对象，可使用下面的语句来实现：

```
对象.Axes(Type,group)
```

参数说明：

- ❑ Type 参数代表坐标轴的类型，其值可以为 xlAxisType 常量之一。当其值为 xlCategory 时，表示坐标轴显示的是数据类别；当其值为 xlSeries 时，坐标轴显示数据系列；当其值为 xlValue 时，坐标轴显示数值。
- ❑ group 参数表示坐标轴组的类型，其值可以是 xlAxisGroup 常量之一。当其值为 xlPrimary 时，表示主坐标轴组；其值为 xlSecondary 时，表示次坐标轴组。

【范例 17-5】 使用 Excel 提示对话框显示图表中纵轴的最大值和最小值，同时判断两个坐标轴是否有标题，若没有，给出提示并添加标题。代码如下：

```
Public sub 坐标轴操作()
    Dim myChart  As Chart                                      '声明变量
    Set myChart = Worksheets(1).ChartObjects(1).Chart          '指定图表
    With myChart.Axes(xlValue)                                 '指定横轴
        If .HasTitle = True Then                               '如果显示标题
            MsgBox "Y 坐标轴标题为：" & .AxisTitle.Text & vbCrLf _
            & "最小刻度为：" & .MinimumScale & vbCrLf _
            & "最大刻度为：" & .MaximumScale        '显示标题和最大值最小值
        Else
            MsgBox "Y 坐标轴没有标题，将添加标题“成绩表”！" & vbCrLf _
            & "最小刻度为：" & .MinimumScale & vbCrLf _
            & "最大刻度为：" & .MaximumScale        '提示并显示最大值和最小值
            .HasTitle = True                        '显示标题
            .AxisTitle.Text = "成绩"                '显示标题文字
        End If
      .AxisTitle.Font.Size = 18                     '设置纵轴标题文字大小
    End With
    With myChart.Axes(xlCategory)                              '指定纵轴
        If .HasTitle = True Then                               '如果显示标题
            MsgBox "X 坐标轴标题为：" & .AxisTitle.Text        '显示标题内容
     Else
            MsgBox "X 坐标轴没有标题， _
    将添加标题“学生”！"                                    '提示添加标题
         .HasTitle = True                                  '显示标题
         .AxisTitle.Text = "学生"                          '标题内容
     End If
         .AxisTitle.Font.Size = 18                         '设置横轴标题文字大小
  End With
End Sub
```

【运行结果】创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行上述代码，如果图表中纵轴没有标题，则提示没有标题，同时显示纵轴的最大值和最小值，如图 17.14 所示。关闭第一个提示对话框后，如果横轴也没有标题，程序给出第二个提示对话框提示，如图 17.15 所示。关闭该对话框后，图表中添加标题，如图 17.16 所示。

【代码解析】本范例主要介绍使用 Axes 对象集设置坐标轴的方法。在程序中，myChart.Axes(xlValue)语句和 myChart.Axes(xlCategory)分别指定了图表的横轴和纵轴显示的内容。HasTitle 属性的值决定了两轴的标题是否存在。MinimumScale 和 MaximumScale 是 Axis 对象的两个属性，分别返回数轴上的最大值和最小值。AxisTitle 对象的 Text 属性决定了标题显示的内容。

警告：与 Chart 对象中标题的设置一样，HasTitle 属性的值必须设置为 True，标题才能显示出来。

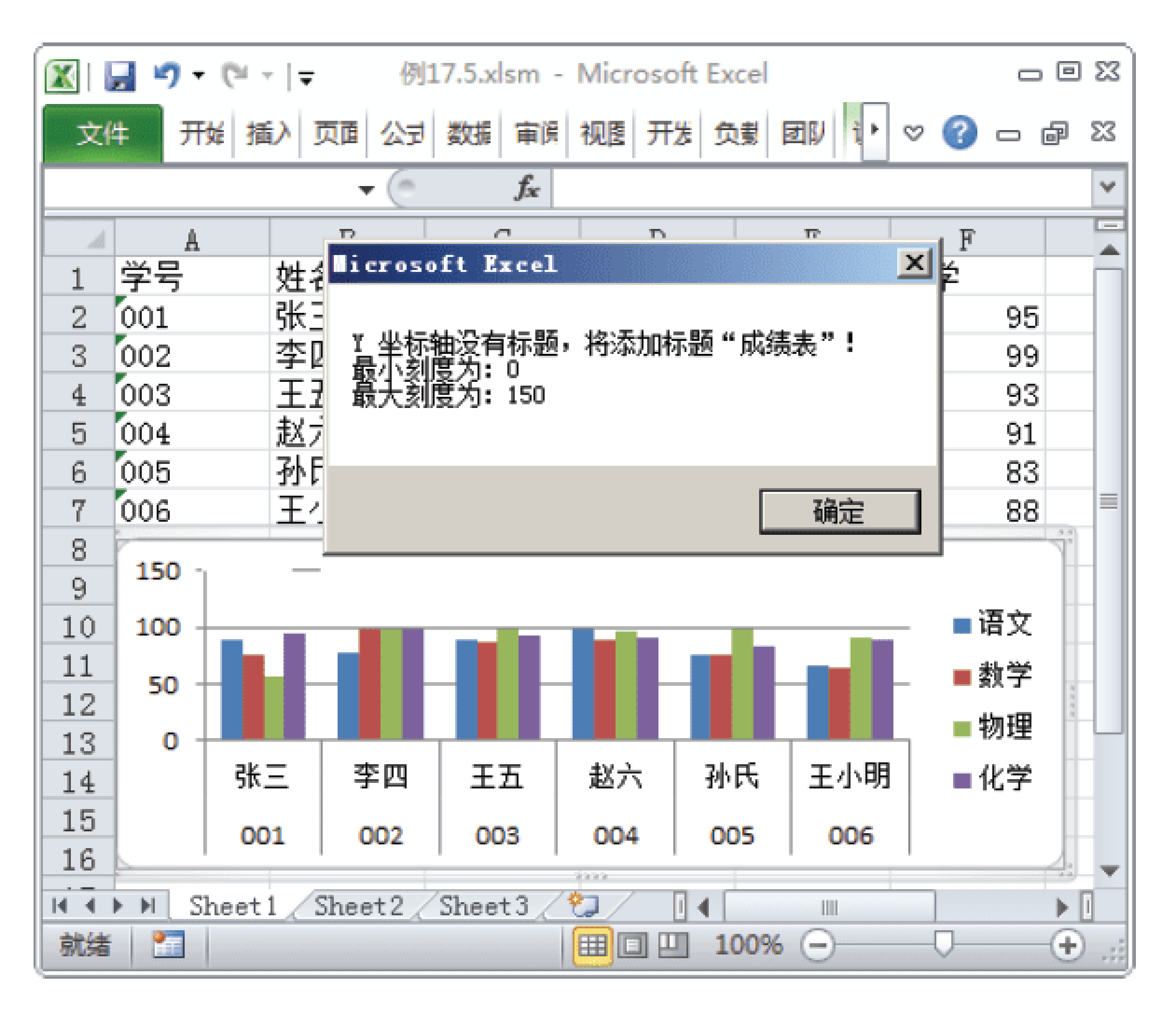


图 17.14　提示纵轴的最大值和最小值以及 Y 轴没有标题

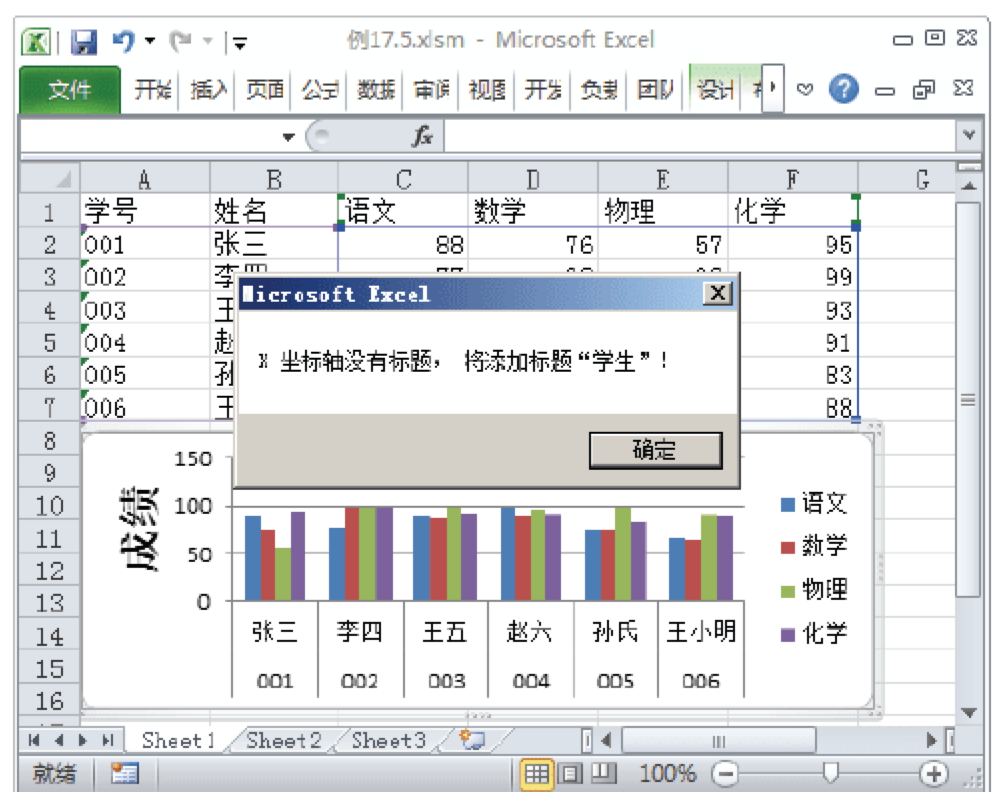


图 17.15　提示横轴没有标题

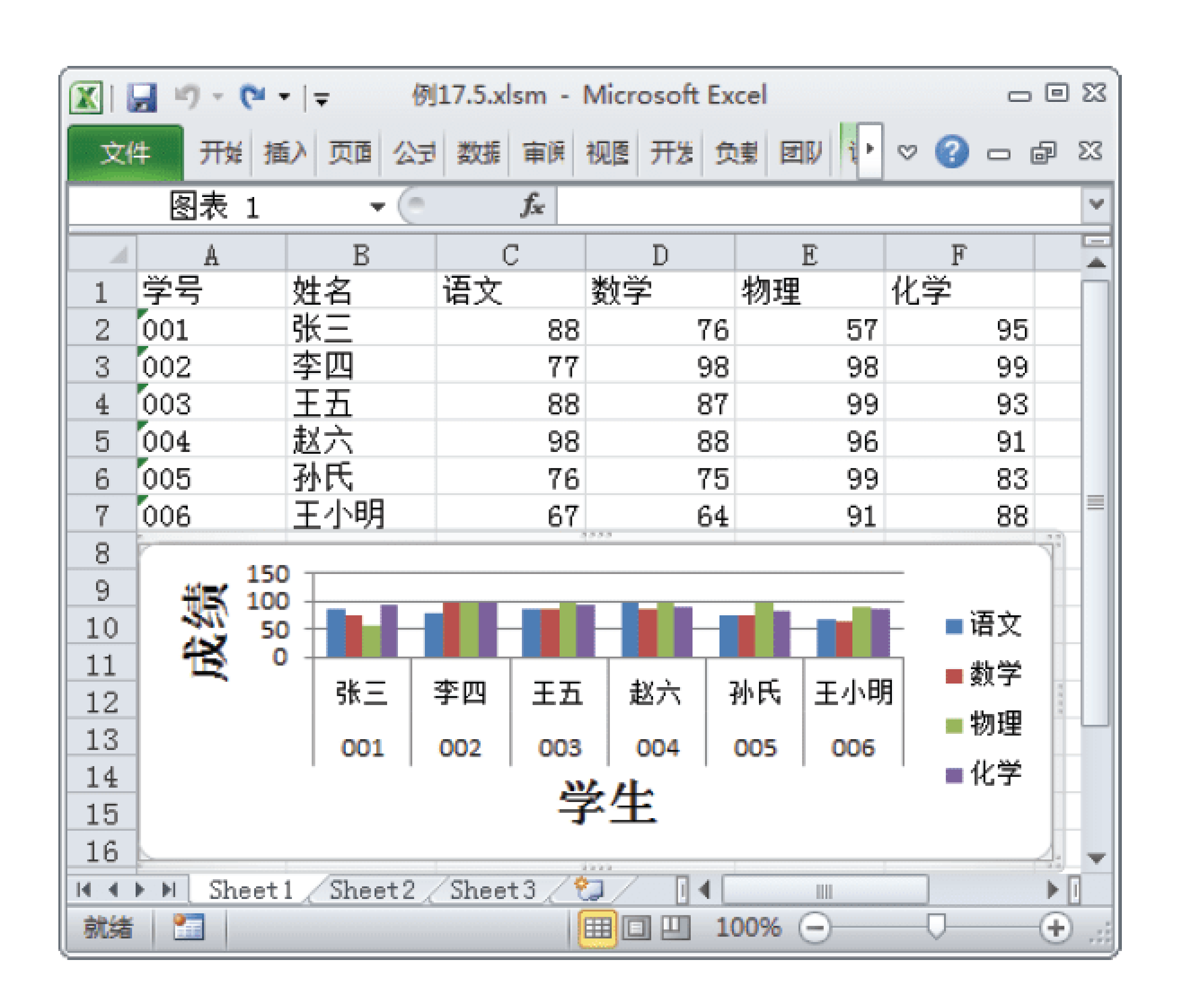


图 17.16　添加标题后的图表

17.3.4　操作数据系列

数据是图表上的重要元素，在 VBA 中，Series 对象代表图表上的数据系列，该对象是 SeriesCollection 集合的一个成员。如果需要指定图表中的一个数据系列，可以使用下面的代码：

```
对象.SeriesCollection(Index)
```

这里，Index 参数是从 1 开始的索引号，其最大值是图表中数据系列的数量或数据系列名称的数量。如果需要查看这个索引号，可以在图表中单击某个数据系列，此时在 Excel 2010 的编辑栏中就可以看到一个公式，如图 17.17 所示。这个公式中最后的那个数字就是数据系列的索引号。

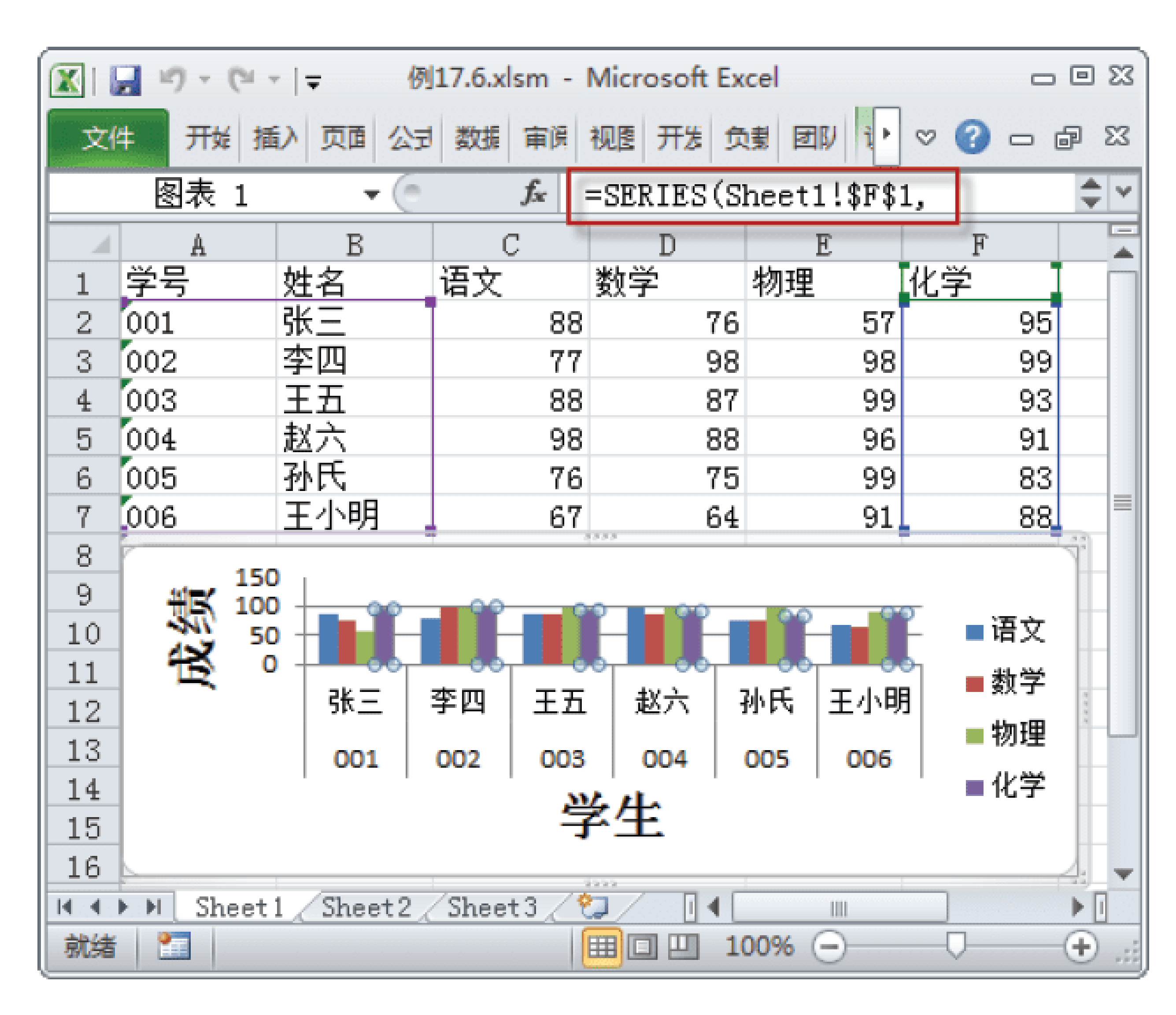


图 17.17　查看索引号

【范例 17-6】 编写程序，在工作表中创建一个折线图，并设置折线图数据系列样式，程序代码如下所示。

```
Sub 设置数据系列()
    Dim n As Integer                                    '声明变量
    For n = 1 To Worksheets(1).ChartObjects(1).Chart. _
    SeriesCollection.Count                              '遍历所有数据系列
        With Worksheets(1).ChartObjects(1).Chart.
        SeriesCollection(n)                             '选择数据系列
            With .Border                                '设置数据系列边框
                .ColorIndex = 3                         '设置边框颜色
                 .Weight = xlThin                       '设置边框宽度
                 .LineStyle = xlDot                     '设置边框线型
              End With
            .MarkerStyle = xlMarkerStyleDiamond         '标记样式为菱形
            .Smooth = True                              '线条平滑
            .MarkerSize = 5                             '设置标记大小
        End With
```

```
16        Next
17    End Sub
```

【运行结果】在 Excel 2010 工作表中首先创建一个折线图，在 Visual Basic 编辑器中创建一个模块，打开该模块的“代码”窗口输入上述代码。按 F5 键运行上述代码，折线图的数据系列样式变为程序设定的样式，如图 17.18 所示。

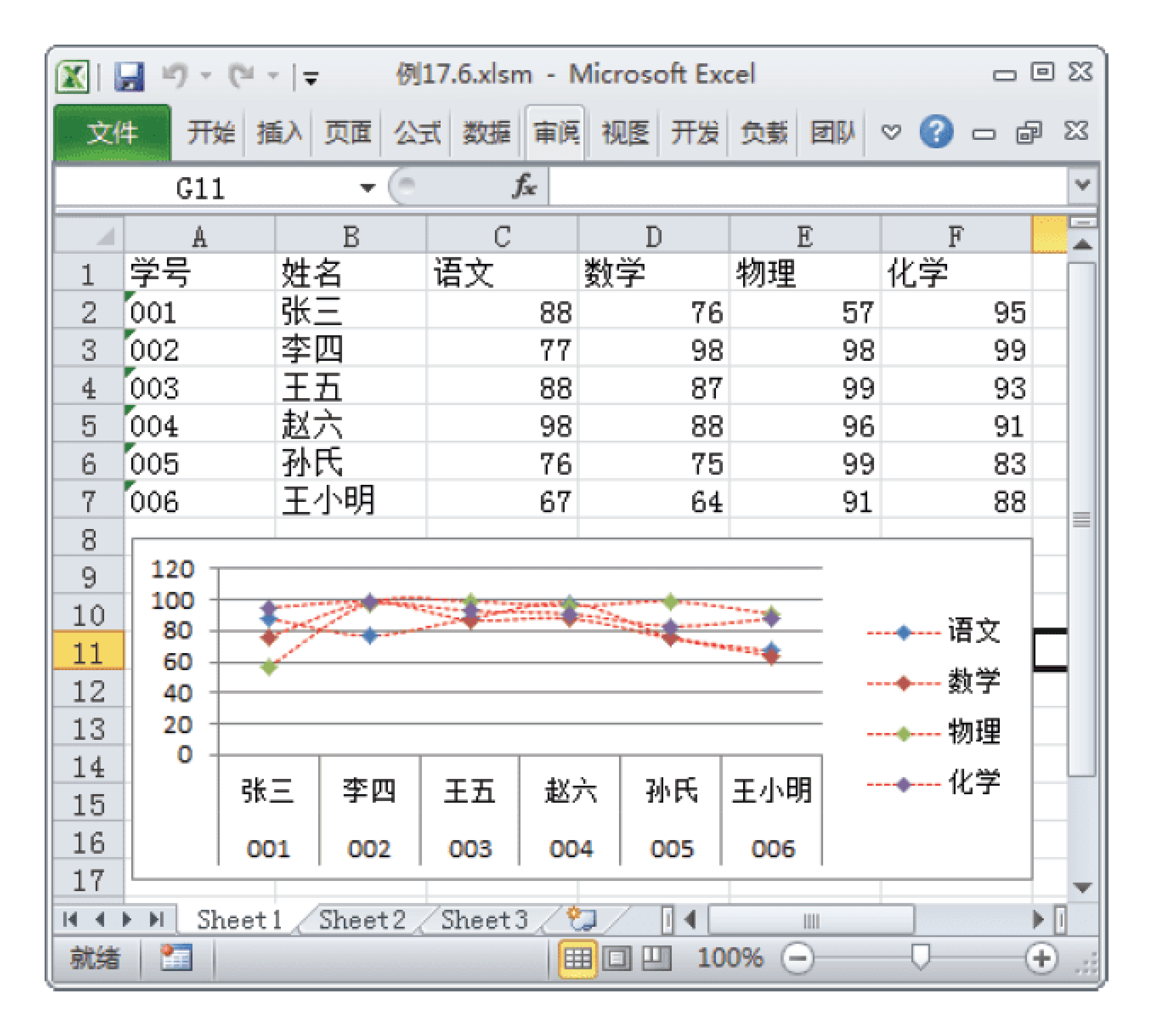


图 17.18　折线图的样式变为程序设定样式

【代码解析】本范例演示修改数据系列的方法。在程序中，Worksheets(1).ChartObjects(1).Chart.SeriesCollection.Count 使用 SeriesCollection 对象的 Count 属性获得数据系列的总数。程序使用 For…Next 循环遍历数据系列中所有对象，分别设置它们的属性。在程序中，第 06~10 行代码用于设置数据系列的边框样式。第 12 行代码使用 MarkerStyle 属性设置折线图上标记的样式，第 14 行代码使用 MarkerSize 属性设置标记的大小，而第 13 行代码将折线设置为平滑线条。

注意：只有折线图、散点图或雷达图中存在数据标记，因此 MarkerStyle 属性和 MarkerSize 属性只能使用于这些图表中，否则程序运行时会出错。

17.4　小　　结

本章学习了使用 VBA 创建图表的方法，学习了使用图表对象的属性、方法和事件来对图表对象编程的方法，同时介绍了对图表的图表区、绘图区、坐标轴和数据系列进行设置的方法。

图表是 Excel 2010 中常见的一种对象，嵌入式图表和图表工作表在引用时的方式是不同的，这是读者必须要注意的问题。同时，在编写程序时，使用对象变量来引用图表是操作图表的好方法。使用对象变量可以在不激活图表的情况下直接引用图表，同时也能简化

代码的编写。图表所涉及对象很多，掌握对象的从属关系以及常见的属性和方法是关键。

17.5　本 章 习 题

1．下面语句的作用是什么？（　　）

```
For Each a In charts("Chart1").Axes
    a.HasMajorGridlines=False
    a.HasMinorGridlines=False
Next
```

A．使图表 Chart1 中的坐标轴不可见

B．使图表 Chart1 中的坐标轴网格线样式不能更改

C．使图表 Chart1 中坐标轴的网格线都不可见

D．使图表 Chart1 中坐标轴不显示主标题和副标题

2．对下面语句描述正确的是哪个选项？（　　）

```
Charts("Chart1").SeriesCollection(1).Value=Worksheets("sheet1").Range
("A1:A6")
```

A．设置图表 Chart1 的数据系列值为工作表 A1 至 A6 单元格的值

B．设置图表 Chart1 的数据系列名称为工作表 A1 至 A6 单元格的值

C．设置图表 Chart1 的数据系列中横坐标值为 A1 至 A6 单元格的值

D．设置图表 Chart1 的数据系列中纵坐标的值为 A1 至 A6 单元格的值

3．选择下面程序的运行结果。（　　）

```
Sub test()
    With Charts("Chart1")
    .HasTitle=True
    .ChartTitle.Text="2008 年财务状况"
End Sub
```

A．图表名设置为“2008 年财务状况”

B．图表的标题设置为“2008 年财务状况”

C．图表的图例名设置为“2008 年财务状况”

D．图表的纵轴标题设置为“2008 年财务状况”

4．选择下面程序的运行结果。（　　）

```
Sub test()
    With ActiveChart
        Do Until.SeriesCollection.Count=0
            .SeriesCollection(1).Delete
        Loop
    End With
End Sub
```

A．获得当前图表的数量　　　　B．删除所有图表

C．清除所有数据系列　　　　　D．清除图表中所有数字

5．编写程序创建一个成绩表，为成绩表中的图表添加“历史”科目分数的数据系列。

【提示】使用 SeriesCollection 对象的 NewSeries 方法能够创建一个新的数据系列。数据系列对象 Series 的 Name 属性能够设置对象名称，Values 对象能够设置数据系列的值，Xvalue 属性可对图表中 X 轴上数据进行设置。

6．编写程序创建一个表格，修改图表字的字体、颜色和大小，并修改图表区边框的线型、宽度和颜色，同时修改图表背景颜色。

【提示】本练习是对图表的图表区中对象的操作。文字的字体、大小和颜色可以分别使用 Font 对象的 Name 属性、Size 属性和 ColorIndex 属性来设置。图表边框的线型、线宽和颜色可以分别使用 LineStyle 属性、Weight 属性和 ColorIndex 属性来设置。而图表背景，可以使用 Interior 对象的 ColorLndex 属性来设置。

第 18 章　类　模　块

类模块向开发人员提供了创建和操作自定义对象类的功能，用户自定义类可以使 VBA 程序设计具有更加模块化，同时使程序设计更能反映实际状况。类具有属性、方法和事件，通过实现类的属性、方法和事件实现对类模块的设计，本章学习的主要内容和目标如下：

- 理解类的基本概念；
- 学习建立对象类、声明类模块中的对象的方法；
- 学习在程序中使用类属性；
- 学习在程序中为类灵活创建方法；
- 学习在程序中使用类事件来触发响应。

18.1　使用对象类

类模块是一种可以插入 VBA 工程中的特殊模块，要使用类就必须在程序中创建类模块，而要灵活使用类必须掌握类中对象声明的方法。

18.1.1　创建对象类

在日常生活中，有很多相同或相似的对象，如计算机上用于输入的键盘和鼠标。在工业生产中，一般都是先完成这个产品的模具，然后通过该模具来进行大量的生产。在程序设计中，往往会遇到相同或相似的对象，在实际编写代码时，同样可以采用与工业产品的生产相同的模式，使用“模具”来实现对这些相同或相似对象的操作。这里，每一个用于生产的“模具”就是类，而获得的产品就是一个对象。

实际上对于类，读者并不陌生，前面章节中学习的对 Excel 工作簿、工作表和单元格的操作实际上都是通过对相应的对象类的操作来实现的，如：

```
Dim mySheet As WorkSheet
```

该语句定义了一个 WorkSheet 对象，Sheet 对象就是 VBA 中 VBA 内部定义好的一个类，这个类称为标准类，如 Workbook 和 Range 均属于标准类。在编写程序时，不管何种工作表，只要使用 WorkSheet 对象的属性、方法和事件，都可以实现对工作表对象的操作。

Excel VBA 为用户提供了大量的标准类，但在实际开发中，用户往往还是需要定义自己的类来完成某些特定的操作。VBA 提供了类模块的功能，允许用户根据需要定义自己的对象类。在 VBA 中，要创建对象类，首先需要在程序中创建一个类模块。下面以创建一个名为 NewClass 类为例来介绍创建类模块的方法。

（1）启动 Excel 2010 创建一个新工作簿。打开 Visual Basic 编辑器，选择“插入”→“类模块”命令插入一个类模块。此时会打开类模块的“代码”窗口，在工程资源管理器中可以看到插入的类模块，如图 18.1 所示。

（2）打开“属性”窗口，在“名称”栏中输入类的名称，如图 18.2 所示。这样，一个名为 NewClass 的类就创建完成，下面就可以对其进一步进行设置和编程以实现其功能。

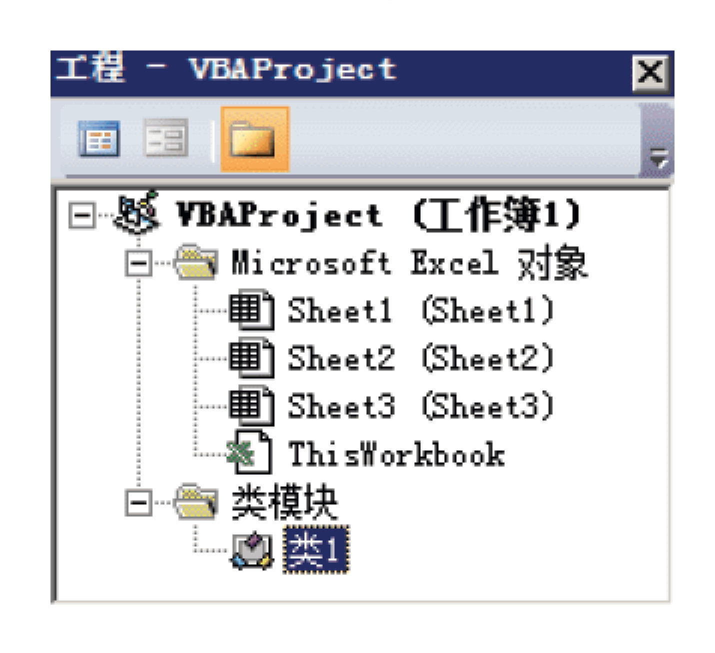


图 18.1　插入一个类模块



图 18.2　输入类名称

提示：在类的“属性”对话框中，Instancing 的设置项决定类在其他 VBA 过程中是否可以被共享。该设置项有 2 个选项，Private 和 PublicNotCreatalbe。当其设置为 Private 时，类中的代码将不允许其他的 VBA 工程访问。当设置为 PublicNotCreatable 时，只有在自己工程中创建了该类的对象才能被其他工程使用。

18.1.2　声明类模块中的对象

创建的类模块与 VBA 标准模块的构件是相同的，都是由变量、过程和函数构成。这些都是标准模块中最基本的构件，同样也是类模块中最基本也是最重要的组成元素。在创建自定义类模块时，对象是由类创建的一个实例，是类的实体化。在创建对象时，一个重要的问题就是考虑对象的作用域和生存周期，其次才是对象的属性和方法。

对象的作用域指的是对象能够在多大的范围内被代码识别。在 VBA 中，当创建类的新实例并赋值给变量后，实例将会存在于内存中，直到该对象变量超过了其作用域为止。在标准模块中，对象变量的作用域取决于声明对象的位置和方式，作用域可以分为过程级、模块级和全局变量这三类。在类模块中，这些作用域的规定同样适用。

注意：在类模块中，这样的作用域划分同样存在。在类模块中，由于类是生成对象的模板，模块级变量在所有生成的各个对象实例间都是相互独立的，当一个对象实例消亡后，伴随它的对象变量也会同样消亡。

在类模块中存在着两类成员，即公有成员和私有成员。在类模块中声明的一组变量、方法、属性和事件定义用户对象的接口。接口由对象成员构成，这些成员可以被外部程序调用，这样的对象成员称为公有成员。在类模块中，如果对象成员不能被外部程序调用，只能供类模块内部的其他过程调用，则其称为私有成员。

例如，在类模块中放置下面语句，该语句将变量 Name 声明为私有变量，则该变量将允

许工程中的类的所有用户访问。类中任何用户都可以获得该变量的值，而在工程外的代码将无法对该变量进行赋值操作。

```
Private Name As String
```

提示：在默认情况下，类模块中的 Sub、Function 和 Property 过程都是公有的，而使用 Dim 语句声明的变量则是私有的。在声明变量和过程时，关键字 Public 表示公有的，而 Private 表示私有。另外，使用 Friend 关键字可以声明 Sub、Function 和 Property 过程，使用该关键字能够从外部访问类模块中的私有成员。

18.2　使用对象属性

属性是对象的特性，大多数对象都拥有至少一个属性，用来存放对象的特征值。在类模块中创建属性的方法有两种，一种就是直接在类模块中声明 Public 变量，作为对象的属性，另一种方法就是使用 Property 过程来定义属性值。下面分别介绍这两种方法。

18.2.1　使用变量创建属性

在类中创建属性，最简单的方法就是向类模块中添加一个公有模块级的变量，这个变量在类模块中起到对象属性的作用。在使用类模块时，向变量赋值就是设置模块的属性值，同时这个属性值也可以通过读取变量而读出。在前面章节中，设置工作表和单元格属性时实际上采用的就是这种模式。

下面通过实例来介绍这种创建属性的方法。该实例将使用公用变量 myNumber 作为新建的类 NewClass 的一个属性，其具体的操作方法如下所示。

（1）按照 18.1 节的方法创建一个名为 NewClass 的类模块。在打开的类模块的“代码”窗口中输入声明公有变量的语句，语句如下所示：

```
Public myNumber As Integer
```

（2）完成上面操作后，类的属性实际上就已经能够在标准模块中使用了。插入一个新模块，在该模块的“代码”窗口中输入变量声明语句，当输入语句的前半部分后，按空格键，在 VBA 的代码提示下拉列表中可以找到新创建的类，如图 18.3 所示。选择该类后获得完整的变量声明语句如下所示：

```
Dim Nu As New NewClass
```

（3）此时可以像使用 Excel 中的对象那样来使用类的这个属性了，即输入“Nu.”后会出现对象的属性提示，如图 18.4 所示。选择该属性后，即可读取属性值或给属性赋值。给这个属性赋值的语句如下所示：

```
Nu.myNumber = 36
```




图 18.3　代码提示中找到新创建的类



图 18.4　对象属性提示

18.2.2　使用属性过程

使用 18.2.1 节的方法来创建类的属性虽然简单，但也有一个缺点，那就是类无法知道属性值何时被外部程序修改，属性值无法被限定在一个有效的范围内。实际上，定义属性还可以使用属性过程来完成。

所谓的属性过程，就是 Property 过程。在使用 Property 过程来定义属性后，外部程序如果修改了属性值，将执行 Property 过程。在该过程中可以通过编写程序代码对设置的值进行检查，将其控制在一个规定的范围内。Property 过程有 3 种形式。

- ❑ Property Let：这类过程用来返回类模块当前的属性值。
- ❑ Property Get：这类过程用来读取类模块的属性值。
- ❑ Property Set：这类过程用来设置对对象的引用，将某个对象赋值给对象属性。

下面通过一个实例来介绍创建类属性的方法。该实例为名为 aclass 的类创建一个名为 sN 的属性。在模块中为该属性赋值，并使用 Print 方法显示赋值后的属性值。同时，该属性值能被设为大于 18 的数，小于 18 的数将设置为 18。实例的具体制作步骤如下所示：

（1）打开 Visual Basic 编辑器，插入一个类模块，将其命名为 aclass。类模块的“代码”窗口中首先声明一个 Public 变量 myN，其代码如下所示：

```
01  Public myN As Integer
```

（2）在“代码”窗口中创建 Property Get 过程，其代码如下所示：

```
01  Public Property Get sN() As Integer
02      sN = myN                                          '变量赋值
03  End Property
```

🔔**提示：**Property Let 过程用来改变类的属性值，其基本构成就是声明部分和主体部分。上面代码的第 01 行是声明部分，其包括了要创建属性名称。显然，上面代码的第 01~03 行之间的代码是主体部分，这里可根据需要添加程序代码以实现某种操作或运算，本实例是给属性赋值。

（3）在“代码”窗口中创建 Property Let 过程，代码如下所示：

```
01      Public Property Let sN(myNu As Integer)
02          If myNu > 18 Then                             '判断值是否大于 18
03              myN = myNu                                '大于则变量赋值
04          Else
05              myN = 18                                  '小于则变量值为 18
```

```
        End If
    End Property
```

提示：Property Get 过程用来获取类的属性值，其声明部分包括两个要素，一个是属性名，这里是 sN。另一个是参数声明，即 myNu As Integer 语句。在类中创建属性时，一个 Property Let 过程对应一个具有相同属性名称的 Property Get 过程。需要创建几个属性，就需要几个这样的成对结构。在过程的主体部分可以添加各种功能代码，如这里就对属性值进行了判断，如果属性值大于 18 则 myN 为获得的属性值，否则其值为 18。

（4）在 Visual Basic 编辑器中添加一个模块，在模块中新建一个过程，该过程为 aclass 类的 sN 属性赋值，并使用 Print 方法在立即窗口中显示属性值，如图 18.5 所示。

```
Sub 类属性设置()
    Dim st As New aclass                        '声明变量
    st.sN = 30                                  '属性值设置 30
    Debug.Print st.sN                           '显示当前的属性值
    st.sN = 15                                  '属性值设为 15
    Debug.Print st.sN                           '显示当前的属性值
End Sub
```



图 18.5　“立即”窗口中显示的属性值

提示：从“立即”窗口中显示的结果可以看出，当 sN 属性赋值为 30 时，使用 st.sN 读取的属性值仍是该值。而当 sN 的值设置为 15 时，由于 Property Let 过程中对属性值进行了判断，小于 18 的数都设置为 18，则读取类的属性值不是 15 而是 18。如果想了解程序的运行流程，可以在“类属性设置”模块中按“F8”键逐语句运行程序，可以看到类属性被读取和调用的过程。

18.3　创建类的方法

对于对象来说，一般都至少拥有一个方法。方法是对象可以执行的操作，用户使用方法来实现对类数据的操作。在类中，方法是 Sub 或 Function 过程，在类中添加过程，即可创建类方法。

下面通过实例创建一个名为“myClass”的类，为该类创建一个名为“WriteTo”的方法，该方法能调用输入对话框，将用户需要输入的文字以设定的样式输入激活的单元格中。本实例的制作步骤如下所示：

（1）在 Visual Basic 编辑器中创建一个类，将类命名为 myClass。打开类的“代码”窗口，输入如下的程序代码：

```
01  Public stName As String                       '声明公有变量
02  Sub WriteTo()                                 '声明过程
03      ActiveCell = stName                       '设置激活单元格的文字
04      With ActiveCell.Font
05          .Name = "黑体"                        '设置激活单元格字体
06          .Bold = True                          '设置文字加粗
07          .ColorIndex = 3                       '设置文字颜色
08      End With
09  End Sub
```

提示： 这段代码首先创建了一个公有变量 stName，该变量作为类属性使用。第 02 行至第 09 行的过程就是创建的类方法。

（2）再创建一个模块，在模块输入代码，代码运行时首先给出提示框要求用户输入文字，输入的文字赋予类属性，然后调用类方法将输入文字写入单元格。程序运行效果如图 18.6 所示。

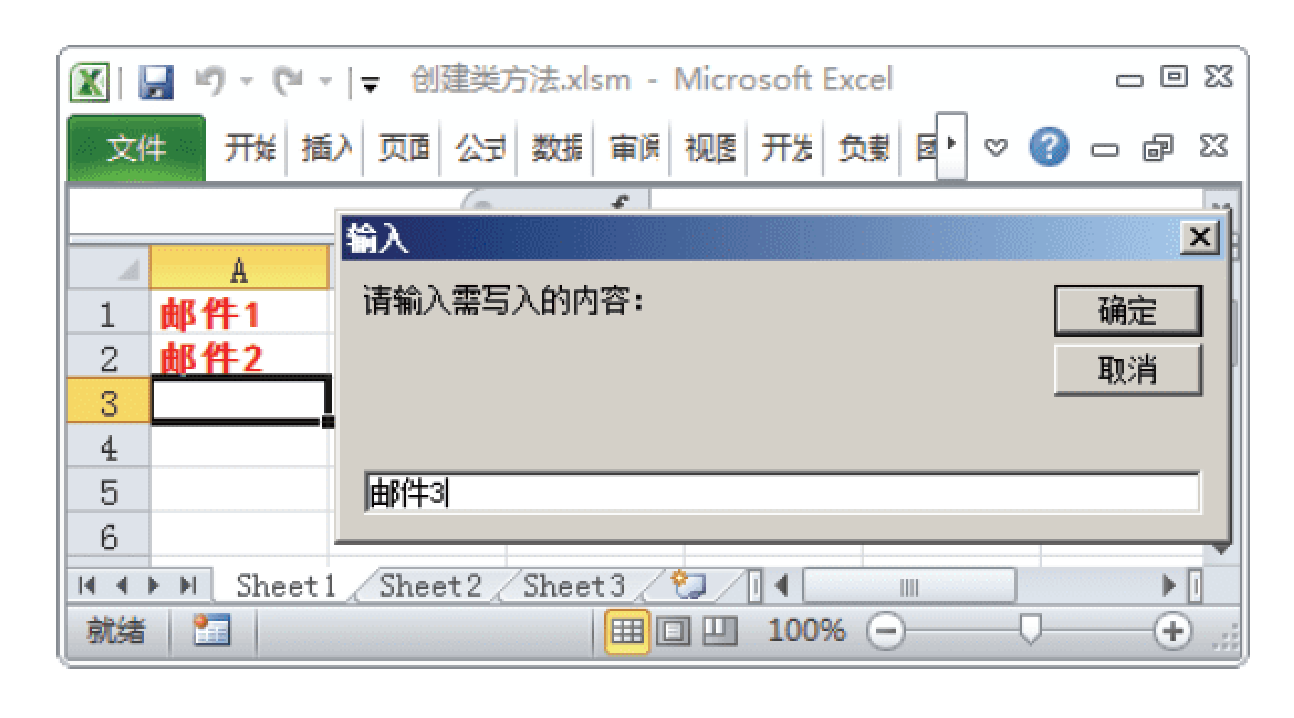


图 18.6　按类方法设置的样式输入文字

具体的程序代码如下所示：

```
01  Sub 类方法创建示例()
02      Dim st As New myClass                              '声明变量
03      st.stName = InputBox("请输入需写入的内容：", "输入")   '设置类变量
04      st.WriteTo                                         '调用类方法
05  End Sub
```

提示： 在这段代码中，使用变量来创建类属性，第 03 行将 InputBox 函数获得的用户输入文字赋予类属性 stName。第 04 行直接调用类中的 WriteTo 过程，该过程就是创建的类的方法。

18.4　类 的 事 件

在前面章节中不只一次地用到了对象的事件，如按钮的单击事件、窗体初始化事件和工作簿的激活事件等。这些对象的事件都是系统预先定义好的。实际上类本身也有两个基本的事件，一个 Class_Initialize 事件（即类加载时触发的事件），另一个 Class_Terminate

（即类卸载时触发的事件）。对于类的使用，更多时候是需要使用创建者为对象自定义的事件。

18.4.1　创建事件的语法

类事件的创建，主要是创建自定义事件和事件响应的过程。创建和使用类事件一般需要经过 3 个步骤：创建事件、触发事件和使用事件过程。首先，要为类定义一个事件，定义事件使用 Event 语句来实现，其语法格式如下所示：

```
[Public] Event procedurename [(arglist)]
```

参数说明：

- Public 是一个可选参数，指定事件在整个工程中可见。此参数可选，在默认情况下，Event 类型是 Public。
- Procedurename 为事件名称，此参数是必须有的。
- arglist 用于声明事件传递的参数。使用 ByVal 表示参数按值传递，使用 ByRef 表示参数按地址传递。

如，定义一个名为 TimerEve 的事件，且事件带有一个按值传递的名为 Jum 的整型参数，此时可以使用下面的语句：

```
Public Event TimerEve(ByVal Jum As Integer)
```

在类模块中创建了事件过程后，如果需要在模块中调用该事件，可以使用 RaiseEvent 语句来触发该事件，其具体的语法格式如下所示：

```
RaiseEvent 事件名称 [(参数列表)]
```

这里，参数列表可选，其可以使用以逗号分隔的变量和数字或表达式列表，参数列表需要用括号括起来，没有参数时括号必须省略。

完成事件的创建后，要在模块或窗体中使用事件过程，需要将对象变量声明为类，声明时使用 WithEvents 关键字。如，将 myTimer 变量声明为 TimerEve 类，可以使用下面的语句：

```
Public WithEvents myTimer As TimerEvent
```

18.4.2　创建事件的案例

下面通过一个实例来介绍类事件的创建和使用方法。在实例中，创建一个类事件，在用户窗体启动时，如果选择单元格的值小于 30，事件触发。事件触发将向用户窗体的文本框中显示提示信息。具体制作步骤如下。

（1）启动 Excel 2010，打开 Visual Basic 编辑器，在工程资源管理器中添加一个类模块，在“属性”对话框中将类模块命名为“myClass”。在类的“代码”窗口中输入如下程序代码：

```
Public Event mSelected()                         '创建事件
```

```
02  Public Sub pqr()                           '声明公有过程
03      If Selection < 30 Then                 '判断第一个单元格值是否小于 30
04          RaiseEvent mSelected               '小于则激活事件
05      End If
06  End Sub
```

提示：在这一段类模块程序代码中，主要完成两个工作。第 01 行，首先创建一个名为 mSelected 的事件。第 02～06 行创建一个对象方法。其中，第 03 行判断单元格的值是否小于 30，如果小于 30，则触发事件。

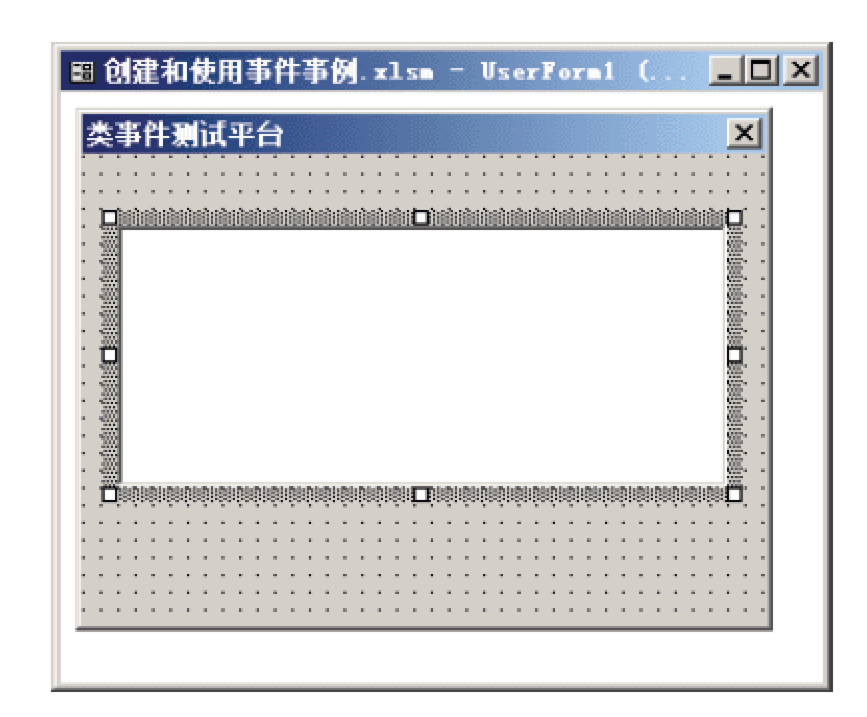
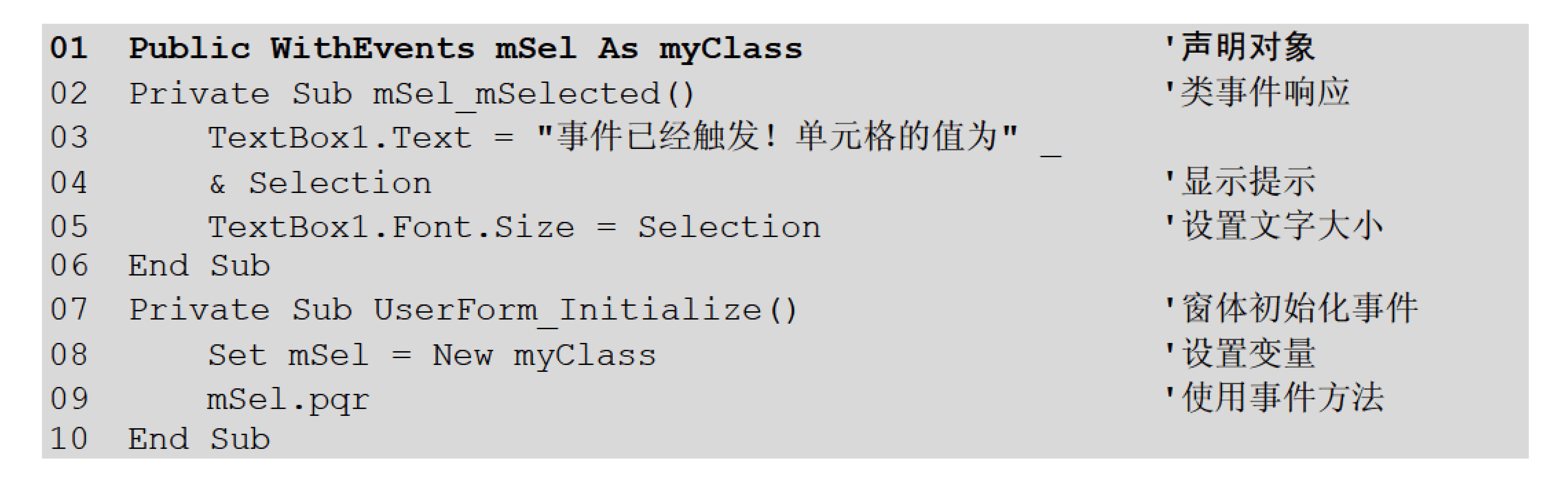


图 18.7　创建用户窗体

（2）在工程资源管理器中右击，在弹出的快捷菜单中选择“插入”→“用户窗体”命令，插入一个用户窗体，在用户窗体中添加一个“文本框”控件。修改用户窗体的标题，如图 18.7 所示。

（3）打开用户窗体的“代码”窗口，为代码添加如下程序代码：



```
01  Public WithEvents mSel As myClass                    '声明对象
02  Private Sub mSel_mSelected()                         '类事件响应
03      TextBox1.Text = "事件已经触发！单元格的值为" _
04      & Selection                                      '显示提示
05      TextBox1.Font.Size = Selection                   '设置文字大小
06  End Sub
07  Private Sub UserForm_Initialize()                    '窗体初始化事件
08      Set mSel = New myClass                           '设置变量
09      mSel.pqr                                         '使用事件方法
10  End Sub
```

提示：本段代码包含了类对象声明、类事件响应代码和窗体初始化代码。这里，第 01 行声明对象变量 mSel 为 myClass 类。第 02～06 行是类事件响应代码，当事件发生时在窗体的文本框中显示单元格的值，同时以单元格的值作为文字大小值。第 07～10 行为窗体初始化，此时指定对象变量，并使用类对象的 pqr 方法。

（4）完成上述制作过程后运行程序，当打开用户窗体时，如果选择单元格中数字小于 30，类事件触发，窗体文本框中显示提示。此时程序运行效果如图 18.18 所示。

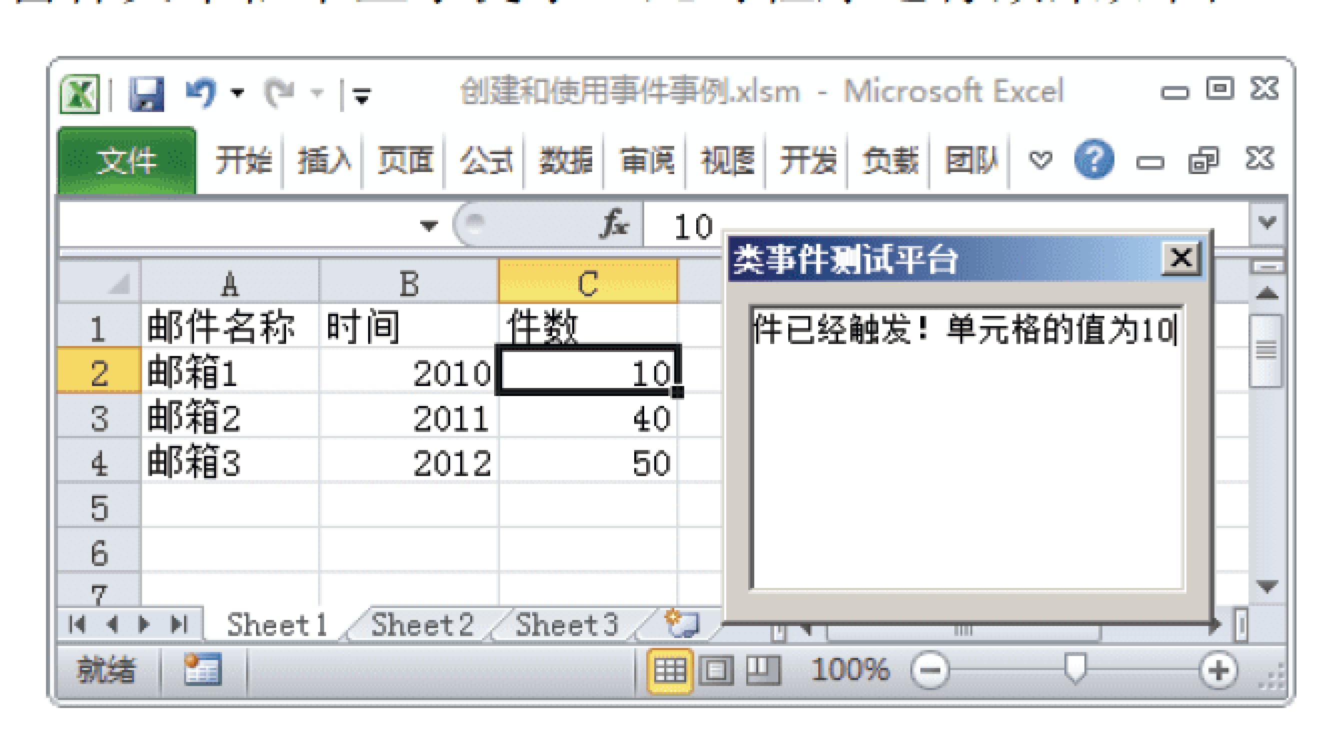


图 18.8　程序运行效果

注意：在类模块中对事件过程进行了定义，在窗体代码中也声明了代表事件过程的对象变量，但事件过程是不会自动运行的，必须添加触发事件的 RaiseEvent 语句，只有其运行了事件才会发生。

18.5 小　　结

类是面向对象编程的一个基本概念，类在程序设计中也起着重要的作用。本章介绍了类的概念及创建类的方法，同时介绍了创建和使用类属性、类方法和类属性的方法和技巧。通过本章的学习，读者将对类有一个更加深入的理解。

在程序中使用类会带来很多便利，从前面学习 VBA 的各种对象就可以体会到这一点。类是编程的高级应用，对于初学者来说，要想掌握类，需要多动手，并且多借鉴别人的优秀的设计思路和设计方法，循序渐进，最终将能够灵活地掌握它。

18.6 本 章 习 题

1．下面语句的作用是什么？（　　）

```
Dim MyName As New NewClass
```

A．创建一个名为 MyName 的类　　B．创建一个名 MyName 的类属性
C．创建一个名为 MyName 的类方法　　D．创建一个名为 MyName 的类事件

2．在类模块中，使用下面哪个关键字将得到私有过程？（　　）

A．Private　　B．Public　　C．Sub　　D．Property

3．创建一个 MyClass 的类，其属性过程如下面代码第 07~09 行所示。运行 test 过程选择其运行结果。（　　）

```
01  Public p As Integer
02  Sub test()
03      Dim s As New MyClass
04      s.n=12
05      Debug.Print s.n
06  End Sub
07  Public Property Let n (n01 As Integer)
08      p=n01-2
09  End Property
```

A．“立即窗口”中显示 12　　B．“立即窗口”中没有显示
C．“立即窗口”显示 0　　D．“立即窗口”显示 10

4．创建一个 MyClass 的类，类过程如代码第 07～09 行所示。运行 Test 过程选择其运行结果。（　　）

```
01  Public S As String
02  Sub WT()
03      ActiveCell = S
04      ActiveCell.Font.ColorIndex = 3
```

```
End Sub
Sub Test()
    Dim st As New myClass
    st.S="王明"
    st.WT
End Sub
```

A．没有任何结果

B．当前选择单元格填入 3

C．当前选择单元格填入“王明”

D．当前选择单元格写入“王明”并且更改颜色

5．创建一个类，使用类查询单元格，当类初始化时，查询工作表中空白单元格的个数并显示结果。

【提示】类的 Initialize 事件在类初始化时发生，显示提示对话框提示工作表中空白单元格的个数。这里可以使用 Range 对象的 SpecialCells 方法来获得与指定类型相匹配的单元格，使用 Count 方法来获得这类单元格的个数。

6．创建一个类，用于筛选工作表中语文成绩大于 60 分的学生。

【提示】创建类，在类中创建一个过程，过程用于成绩筛选。创建一个模块，在模块中将对象赋予一个对象变量，同时调用类中的过程即可实现功能。

第 19 章　数据库编程

数据库在程序设计中具有举足轻重的地位，数据库由多个数据表构成，每个数据表由多条数据记录构进。VBA 通过 ADO 对象可以连接数据、插入记录、删除记录，且可以利用 SQL 语言实现对数据的查询，并将数据从数据库中提取出来显示在 Excel 表格中或者用户自定义的界面上。本章学习的内容和目标如下：

- 理解数据库的概念，掌握数据库的构成；
- 掌握使用连接对象连接数据库的方法；
- 掌握使用添加、删除数据记录的方法；
- 掌握 SQL 查询语言，能够使用查询语言从数据库中筛选需要的记录。

19.1　认识数据库

通俗地说，数据库就是存储数据的仓库，是一种长期存储在计算机内的、有组织的并且可以共享的数据集合。数据库中的数据是按照一定的数据模型进行组织、描述和储存的，可以为各种用户共享。常见的数据库类型包括 Access、SQL Server、FoxPro、Oracle 以及 ASCII 文本文件等。

要想使用数据库，首先必须了解数据库的构成。数据库首先是由数据表（Table）构成，数据表指的是数据库中相同类型数据记录所组成的集合。一个数据库中可以有许多的数据表，如学校的成绩管理数据库可以包含年级的成绩表、各个班级的成绩表以及各个班级各个科目的成绩表，如图 19.1 所示。不同的数据库软件对数据库及成绩表的处理方式是不同的，有的将数据表置于同一个文件中，如 Access；有的将数据表分置于不同的文件中，如 FoxPro。

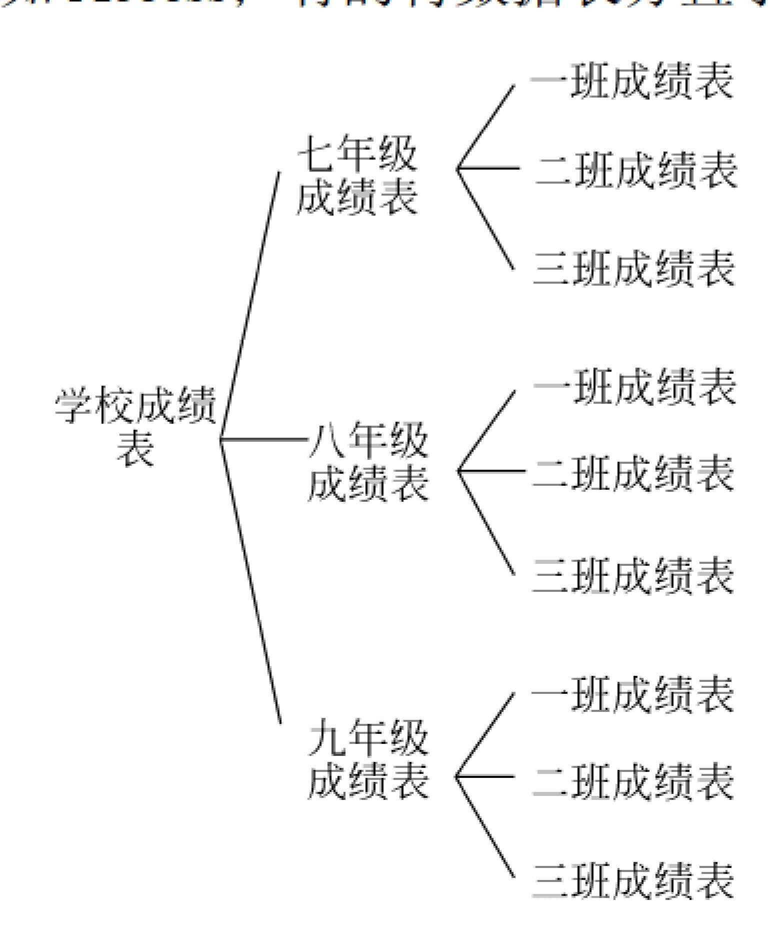


图 19.1　数据库结构模拟图

每个数据表都可以看成是一个二维的数据表，由若干行（Row）和列（Column）构成。每一个列就是一个字段（Field），每个字段都有名称，称为字段名（FieldName）。数据表中每一行（Row）为一个记录，记录是真正的数据主体。

Access 数据表是比较常用的数据库，其结构能够反映刚才介绍的数据库结构。打开一个 Access 数据库，其结构如图 19.2 所示。没有记录、只有字段定义的数据表是一个空的数据表。进行数据的存取，就是针对记录进行一系列的动作，如添加记录和删除记录。

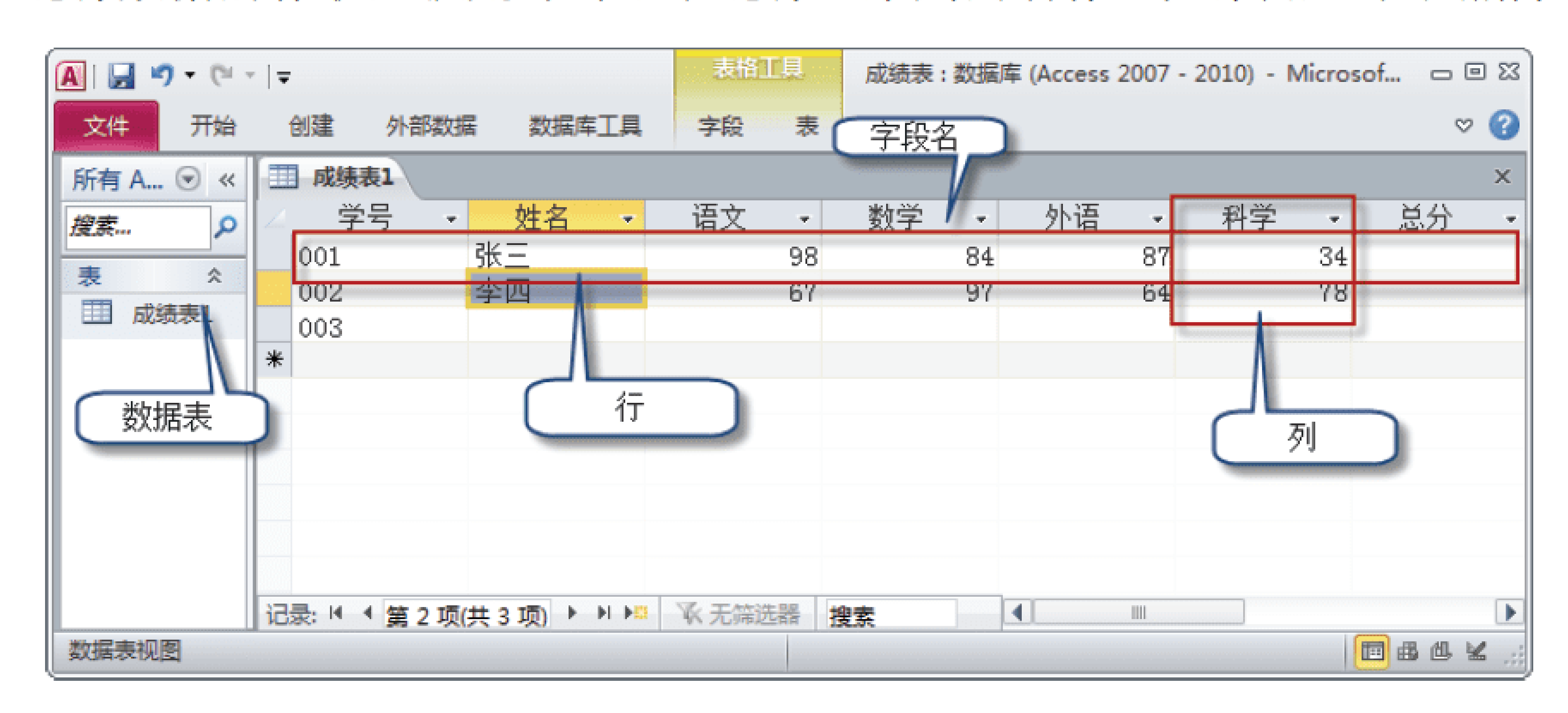


图 19.2　数据表的结构

数据的检索，是操作数据库的常见操作。使用索引是提高数据检索能力的有效手段。索引是根据数据表中关键字字段（KeyField）建立的，这个关键值由一个或多个字段构成。在表中可以没有任何索引，也可以只拥有一个索引或包含多个索引，每个索引由不同的关键字生成。

注意： 对数据进行处理，可以使用数据库来实现，当然也可以使用 Excel。Excel 适用于存储较小的数据，尽管在 Excel 2010 中，工作簿的列达到了 16 384 个，行增加到了 1 048 576 个，单元格数量达到了 1.7×10^{10} 个，但与数据库相比，其存储容量还是偏小。因此对于需要存储大量重复数据的场合，数据库仍然是首选。

19.2　使用 ADO 操作数据

微软公司提供了访问外部数据库的最新方法，那就是使用 ADO（ActiveX Data Objects，即 ActiveX 数据对象）数据对象模型，其被设计用来提供通用数据访问。本节将对 ADO 对象模型以及使用 ADO 访问数据库的基本方法进行介绍。

ADO 的对象模型定义了一个可编程的分层对象集合，其包括的主要组件是：Connection（连接）对象、Command（命令）对象、Recordset（记录集）对象和 Field（字段）对象，这些对象都分别拥有自己的属性集合。

19.2.1　创建 Connection 连接对象

ADO 对象模型中最高级别的对象是 Connection 对象，该对象表示数据源的连接。通

过该对象可以从应用程序访问数据源。在 Excel VBA 中使用 ADO 访问数据库，首先需要设置对 ADO 的引用，具体的方法是：

在 Visual Basic 编辑器中选择“工具”→“引用”命令，打开“引用”对话框，在对话框中选择“Microsoft ActiveX Data Objects 6.1”选项后，单击“确定”按钮关闭对话框，即可在程序中使用 ADO 了，如图 19.3 所示。

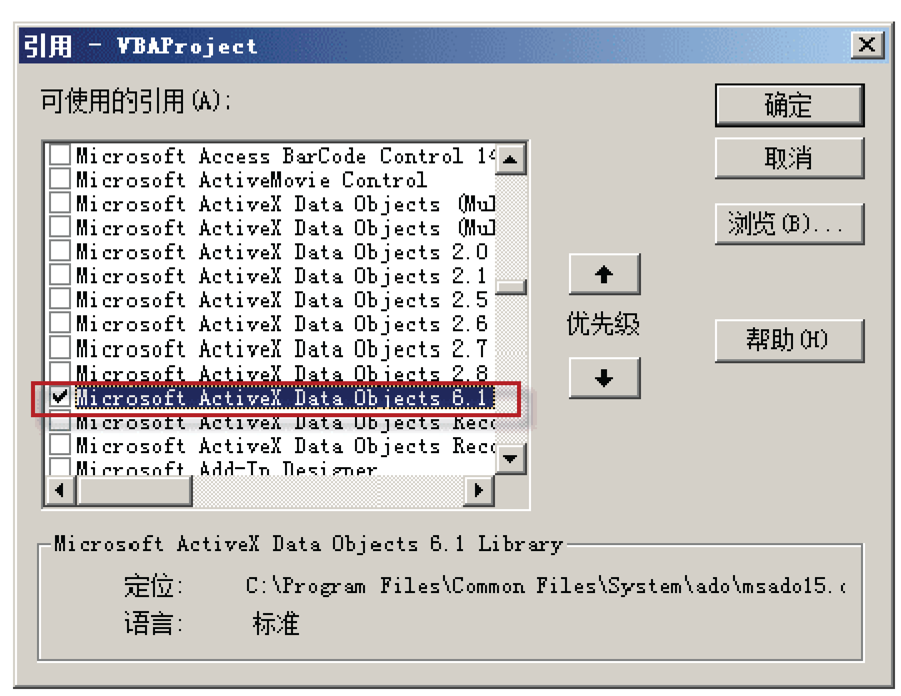


图 19.3 “引用”对话框

在程序中使用 ADO 访问数据时需要通过创建 Connection 对象来建立与数据库的连接。建立与数据库的连接实际上就是在程序中声明一个对象变量，并对其设置具体的引用。如，在下面的语句中，首先声明一个名为 cn 的对象变量，然后使用 Set 语句创建变量与数据库的连接。具体的程序代码如下所示：

```
Dim cn As ADODB.Connect
Set cn=New ADODB.Connect
```

在完成数据源的连接后，需要设置 Connection 对象的属性以指定数据源的类型和 OLEDB 提供者的基本信息，即需要初始化参数。参数的初始化如下面语句所示：

```
Cn.Provider="Microsoft.Jet.OLEDB.4.0
```

提示：这里，初始化参数主要包括 Provider 和 ConnectionStirng 参数。Provider 指出要连接的 OLE DB 提供者的名字。其中，Microsoft Excel Driver(*.xls)表示提供者是 Excel 数据库。Microsoft.Jet.OLEDB.4.0 表示提供者是 Access（2003 及以前版本）mdb 数据库。Microsoft.ACE.OLEBD.12.0 表示提供者是 Access2010 数据库。SQLOLEBD 表示提供者是 SQL Server 数据库。ConnectionString 是个字符串，其包含连接数据源需要的信息。

在指定了 OLEDB 数据提供者和传递了连接信息后，就可以使用 Open 方法来打开数据连接，也就是正式建立连接了。Open 方法的语法规则如下所示：

```
表达式.Open 数据源
```

如，可以使用下面语句来初始化参数并建立与当前活动的 Excel 工作簿的连接：

```
cn.Open "Provider=Microsoft Jet OLEDB.4.0;Extended Properties=Excel
8.0;Data Source=ActiveWorkbook.FullName
```

19.2.2　使用 Recordset 记录集对象

在数据库中，经常需要对一个表或多个表中的数据进行查询，查询返回的记录集合称为是记录集。在 VBA 中，使用 Recordset 对象来访问数据库表或 SQL 查询返回的记录、对已有的记录进行修改、添加和删除等操作。

Recordset 对象可以是一条满足条件的记录，也可以是一组满足条件的记录，使用

Recordset 对象的属性和方法操作来自提供者的数据记录，这些操作包括记录的浏览、记录的修改、记录的添加和记录的删除等。

在建立与数据库的连接后，需要创建一个记录集变量，并且指定该变量引用的具体记录集。如，下面语句声明一个 ADODB 记录集对象，并使用 Set 语句使变量能够引用这个对象：

```
Dim Rds as ADODB.Recordset
Set Rds= New ADODB.Recordset
```

在记录集返回从数据库中查询的结果集后，有两种方式获得这个记录集，一种是使用 Recordset 的 Open 方法，另一个是使用 Connection 对象的 Execute 方法。使用 Open 方法的语法格式如下所示：

```
Recordset.Open Source,ActiveConnection,CursorType,LockType,Options
```

参数说明：

- ❑ Source 参数可以是为变量名、SQL 语句、表名或存储的过程调用等。
- ❑ ActiveConnection 参数为连接数据库的连接字符串。
- ❑ CursorType 参数确定提供者在打开 Recordset 时使用的记录指针类型。当其值为 0 时，即 adOpenForwardOnly，表示使用仅向前类型的指针；当其值为 1 时，即 adOpenKeyset，表示使用键集类型指针，此值为默认值；当其值为 2 时，即 adOpenDynamic，表示使用动态类型指针。当其值为 3 时，即 adOpenStatic，表示使用静态类型指针。
- ❑ LockType 参数决定提供者 Recordset 时使用的锁定类型。当其值为 1 时，即 adLockReadOnly，表示只读，工作表将不能改变数据，此值为默认值；当其值为 2 时，即 adLockPessimistic，表示保守式锁定，在编辑时立即锁定数据源的记录；当其值为 3 时，即 adLockOptimistic，表示开放式锁定，只在调用 Update 方法时才锁定记录；当其值为 4 时，即 adLockBatchOptimistic，表示开放式批更新，其用于批更新模式。
- ❑ Options 参数用于指示提供者如何计算 Source 参数，也可设置从以前保存的 Recordset 文件中恢复 Recordset。

使用 Connect 对象的 Execute 方法的语法格式如下所示：

```
Set record=connection.Execute (CommandText,RecordsAffects,Option)
```

参数说明：

- ❑ CommandText 参数指定需要执行的 SQL 语句、表名、存储过程或特定的提供者文本。
- ❑ RecordsAffects 参数返回操作所影响的记录的数目。
- ❑ Options 参数指示提供者应该如何为 CommandText 参数赋值。

操作数据库的重要内容就是操作记录集，常见的操作就是对记录的读取、删除、添加和修改。在 Excel 中常需要对整个数据集进行读取，然后将记录写到工作表的指定位置。完成这项工作最简单的方法就是使用 CopyFromRecordset 方法来复制整个记录集，该方法的语法结构如下所示：

```
表达式.CopyFromRecordset(Data, MaxRows, MaxColumns)
```

参数说明：

- ❑ Data 用于指定复制到单元格中的 Recordset 对象。
- ❑ MaxRows 用于指定复制到工作表的最大记录数。
- ❑ MaxColumns 用于指定复制到工作表中的最大字段数。

注意：如果省略 MaxRows 参数，表示复制 Recordset 对象中的所有记录。同样地，MaxColumns 参数也可以省略，如果省略，表示复制 Recordset 对象中的所有字段。

打开的对象和记录集在结束操作后应该从内存中删除，也就是关闭对象和记录集。要关闭记录集，可以使用 Recordset 的 Close 方法，然后将记录设置为 Nothing。如，关闭名为 rs 的记录集对象，可以使用下面的语句：

```
rs.Close
Set rs=Nothing
```

如果需要结束一个变量名为 cn 的连接对象，可以使用下面的语句：

```
cn.Close
Set cn=Nothing
```

下面通过一个实例来介绍创建 Recordset 对象的方法。该实例中，首先创建一个 Access 数据库，然后编程将库中所有数据复制到当前工作表中。具体的操作步骤如下所示：

（1）启动 Access 2010，创建一个新的数据库，如图 19.4 所示。保存数据库后，关闭 Access 2010。

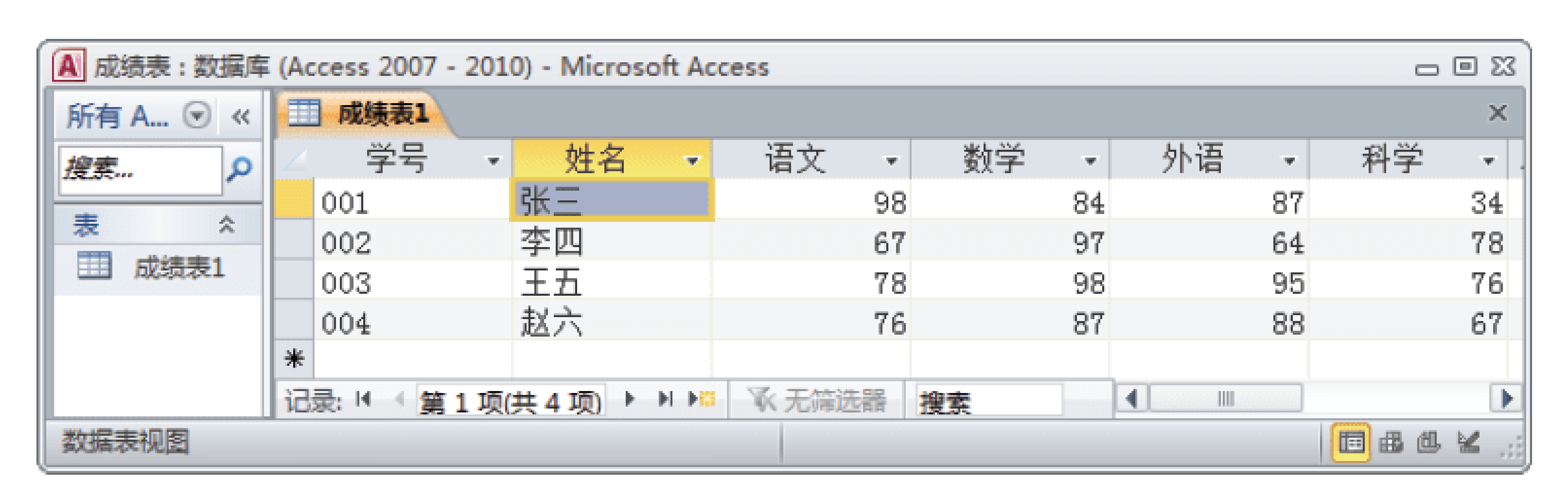


图 19.4　创建 Access2010 数据库

（2）启动 Excel 2010，按照成绩表的字段创建列标题，如图 19.5 所示。打开 Visual Basic 编辑器。选择“工具”→“引用”命令，打开“引用”对话框，在对话框中选中“Microsoft ActiveX Data Objects 6.1”复选框，如图 19.6 所示。单击“确定”按钮，关闭“引用”对话框，完成 ADO 引用设置。

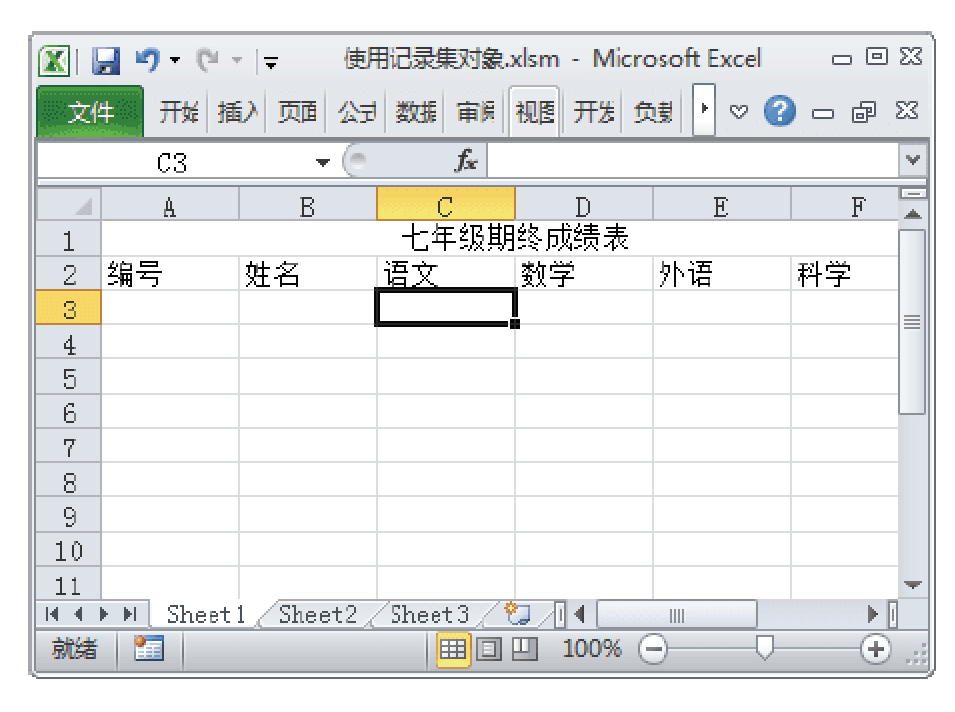


图 19.5　创建工作表标题

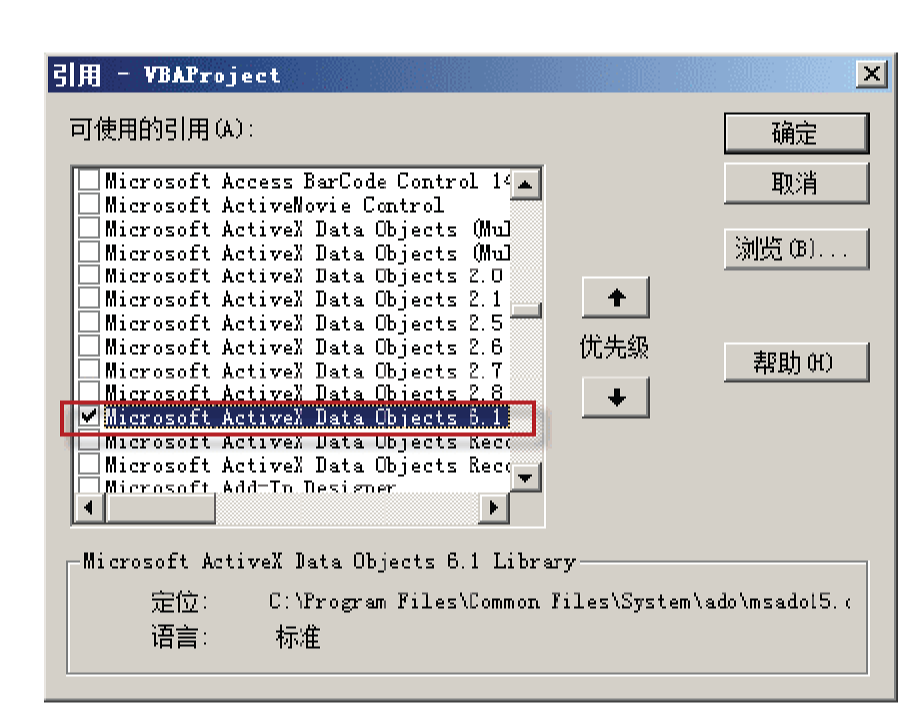
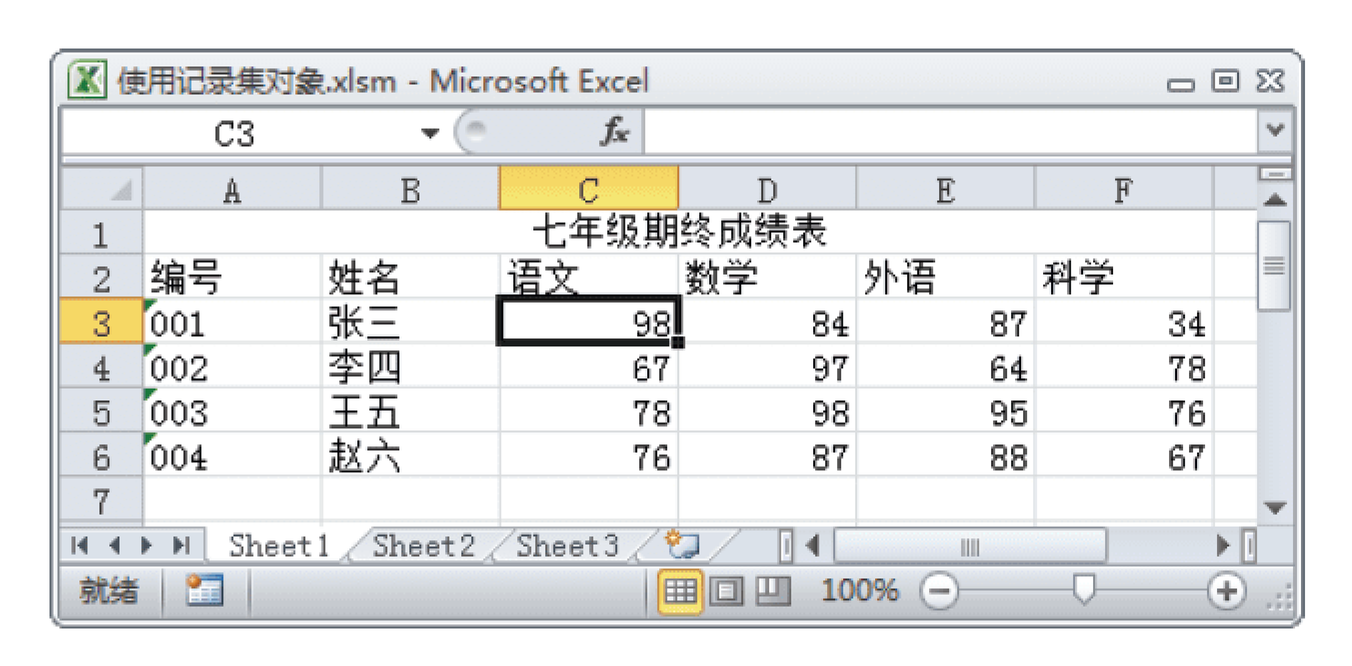


图 19.6　“引用”对话框

（3）创建一个模块，在模块中输入程序代码，代码将 Access 数据库中的数据复制到工作表中，如图 19.7 所示。



	A	B	C	D	E	F
1	七年级期终成绩表					
2	编号	姓名	语文	数学	外语	科学
3	001	张三	98	84	87	34
4	002	李四	67	97	64	78
5	003	王五	78	98	95	76
6	004	赵六	76	87	88	67
7						

图 19.7　复制数据库数据到当前工作表

具体的程序代码如下所示：

```
01  Sub 打开记录()
02      Dim cn As ADODB.Connection                        '声明连接对象
03      Dim rs As ADODB.Recordset                         '声明记录集对象
04      Set cn = New ADODB.Connection                     '创建连接对象变量
05      Set rs = New ADODB.Recordset                      '创建记录集对象变量
06      With cn
07          .Provider = "microsoft.ace.oledb.12.0"        '指明 OLEDB 提供者
08          .Open ThisWorkbook.Path & "\成绩表.accdb"     '打开指定数据库
09      End With
10      rs.Open "成绩表1", cn                          '打开数据库的“成绩表”表单
11      Sheet1.Range("A3").CopyFromRecordset rs        '将表单内容复制到单元格中
12      rs.Close                                       '关闭记录集
13      Set rs = Nothing                               '设置记录为 Nothing
14      cn.Close                                       '关闭连接对象
15      Set cn = Nothing                               '将连接对象设置为Nothing
16  End Sub
```

提示： 上述代码反映了使用 ADO 访问数据库的一般步骤。首先声明 Connection 和 Recordset 变量，就像程序的第 02～05 行那样。然后如第 06～10 行那样打开数据库，在打开数据库时需要指明数据提供者。打开数据库后，再使用 Recordset 对象的 Open 方法打开具体的表单，如第 11 行的语句。只有打开了数据库和表单，才能实现对数据库的操作，这两步缺一不可。在数据库操作完成后，第 13～16 行代码关闭记录集后连接对象。

19.2.3　获取数据库中的数据

数据库中数据的操作，包括数据的引用、对数据库中数据的浏览以及数据的筛选等。记录是由字段构成的，对记录的操作是基于字段的操作。按记录读取数据，实际上就是依次读取数据库中每个字段的数据。在 Recordset 对象中包含一个由 Field 对象组成的 Fields 对象集，其中的每个 Field 对象对应于 Recordset 的一个列。如有一个名为 myField 的字段变量，要获得该字段的标题，可以使用下面语句：

```
myField.Name
```

在记录集中要引用字段有两种方式，即使用字段名称和使用索引号。如，引用 rs 记录集中名为“职称”的字段，使用下面两种语句均可：

```
rs.Fields("职称")
```

或

```
rs! 职称
```

注意：这两种引用方式都是使用字段名称的引用。其中，后面这种方式是一种使用字段名的直接引用，采用这种方式将能获得更快的执行速度，直接引用 rs 记录集中指定的字段。在引用时，记录集后面添加“！”，且字段名称前面必须要带上空格。

字段的索引号是从 0 开始的，使用索引号引用字段，可以使用下面的语句来实现：

```
rs.Fields(0).Value
```

使用 Recordset 对象的 Filter 属性可以选择性地屏蔽 Recordset 对象中的记录，以显示符合条件的记录，使用该属性可以实现精确筛选和模糊筛选。在进行精确筛选时，使用“字段名称＝筛选条件”这种方式来进行筛选。如，在数据库中，筛选字段名为“姓名”的“李飞”记录，可以使用下面的语句来实现：

```
rs.Fileter="姓名"='李飞'"
```

在大多数情况下，筛选的条件不唯一，使用 Like 语句，可以筛选同时满足多个条件的记录。如，在数据库中筛选字段名为“姓名”中最后一个字是“明”字的记录，可以使用下面的语句来实现：

```
rs.Fileter="姓名 Like *'明'"
```

注意：如果取消筛选，使记录恢复到初始状态，可以将 Filter 属性值设置为 AdFilterNone。

下面通过一个实例来进一步熟悉获取数据库数据的方法。本实例的数据库使用上一节中的 Access 数据库。程序运行时，在用户窗体的 2 个文本框中分别输入需要筛选的科目和该科目分数的大于的值，单击“获取”按钮，符合条件的记录写入 Excel 2010 数据表中。下面介绍具体的制作步骤。

（1）启动 Excel 2010，创建一个工作表，添加与 19.2.2 节中工作表相同的表头和标题。打开 Visual Basic 编辑器，创建一个用户窗体。在用户窗体中放置 2 个“文本框”控件、3 个“标签”控件和 2 个“命令”按钮控件，分别设置这些控件的属性，并调整它们在窗口中的位置。完成窗体设计后的控件布局如图 19.8 所示。

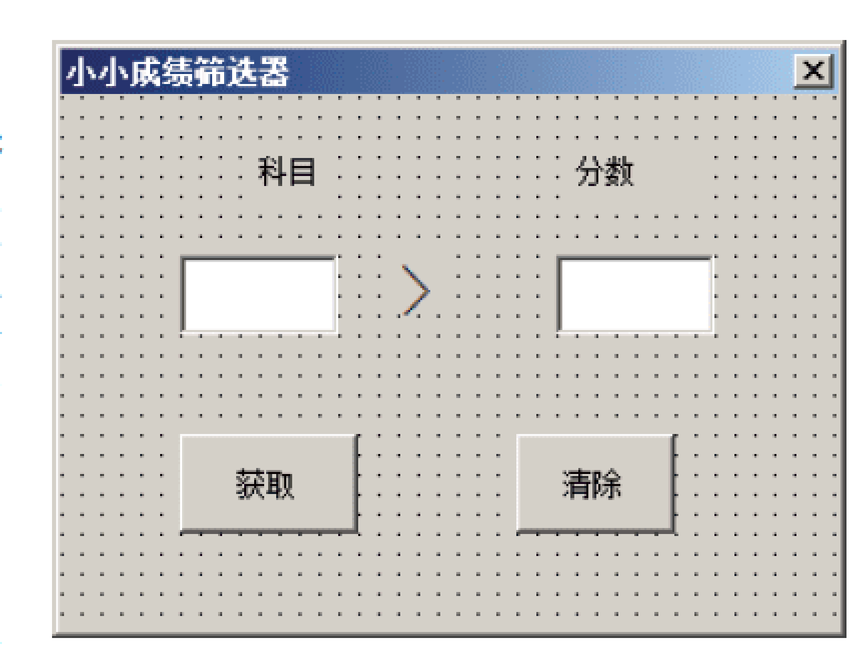


图 19.8　窗体中的控件布局

（2）使用前面介绍的方法完成 ADO 的引用设置。首先为“获取”按钮添加 Click 事件代码。这段代码是实例的关键，代码完成 Recordset 对象的创建，打开数据库，

并根据文本框的输入对数据库进行筛选，筛选结果填入工作表中。“获取”按钮的事件响应代码如下所示：

```
01 Private Sub CommandButton1 Click()
02    Dim fl As Field                                  '声明字段对象
03    Dim n As Integer, a As String, b As String
04    Dim cn As ADODB.Connection                       '声明连接对象
05    Dim rs As ADODB.Recordset                        '声明记录集对象
06    Set cn = New ADODB.Connection                    '创建连接对象变量
07    Set rs = New ADODB.Recordset                     '创建记录集对象变量
08    With cn
09        .Provider = "microsoft.ace.oledb.12.0"       '指明 OLEDB 提供者
10        .Open ThisWorkbook.Path & "\成绩表.accdb"     '打开指定数据库
11    End With
12    rs.Open "成绩表 1", cn                            '打开数据库的"成绩表"表单
13    a = TextBox1.Text                                '获取文本框内容赋予变量
14    b = TextBox2.Text
15    If TextBox1.Text <> "" And TextBox2.Text <> "" Then  '文本框不为空
16        rs.Filter = a & " > " & " '" & b & " '"      '根据输入进行筛选
17    Else
18        MsgBox "请在文本框中输入科目和分值！"           '提示输入
19        TextBox1.SetFocus                          '    文本框获得焦点
20    End If
21    Sheet1.Range("A3").CopyFromRecordset rs          '将筛选结果复制到单元格中
22    rs.Close                                         '关闭记录集对象
23    Set rs = Nothing                                 '将记录对象设置为Nothing
24    cn.Close                                         '关闭连接对象
25    Set cn = Nothing                                 '将连接对象设置为Nothing
26 End Sub
```

提示：这段代码第 02～13 行与 19.2.2 节实例相同，要操作数据库，其打开数据库中表单的模式都是一样的。代码的第 15 行以 2 个文本框的值是否为空作为判断依据，来判断是否输入了科目和分数。第 17 行代码使用 Filter 方法对数据库进行筛选。在程序的第 22 行将筛选结果写入指定单元格。这里值得注意的是，如何将符合条件的筛选结果写入 Excel 数据表，使用的是 CopyFromRecodset 方法，筛选结果在 rs 对象变量中。

运行这段代码，输入筛选条件，数据筛选后的效果如图 19.9 所示。如果单击“获取”按钮时，单元格为空，则程序给出提示，如图 19.10 所示。

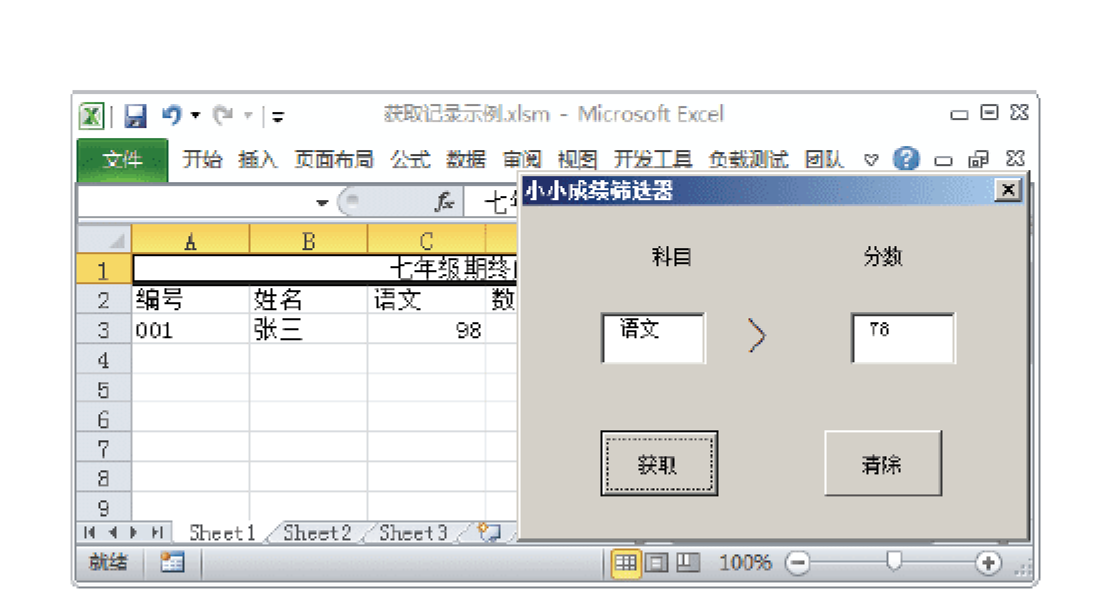




图 19.9　出现的对话框以及数据筛选的结果

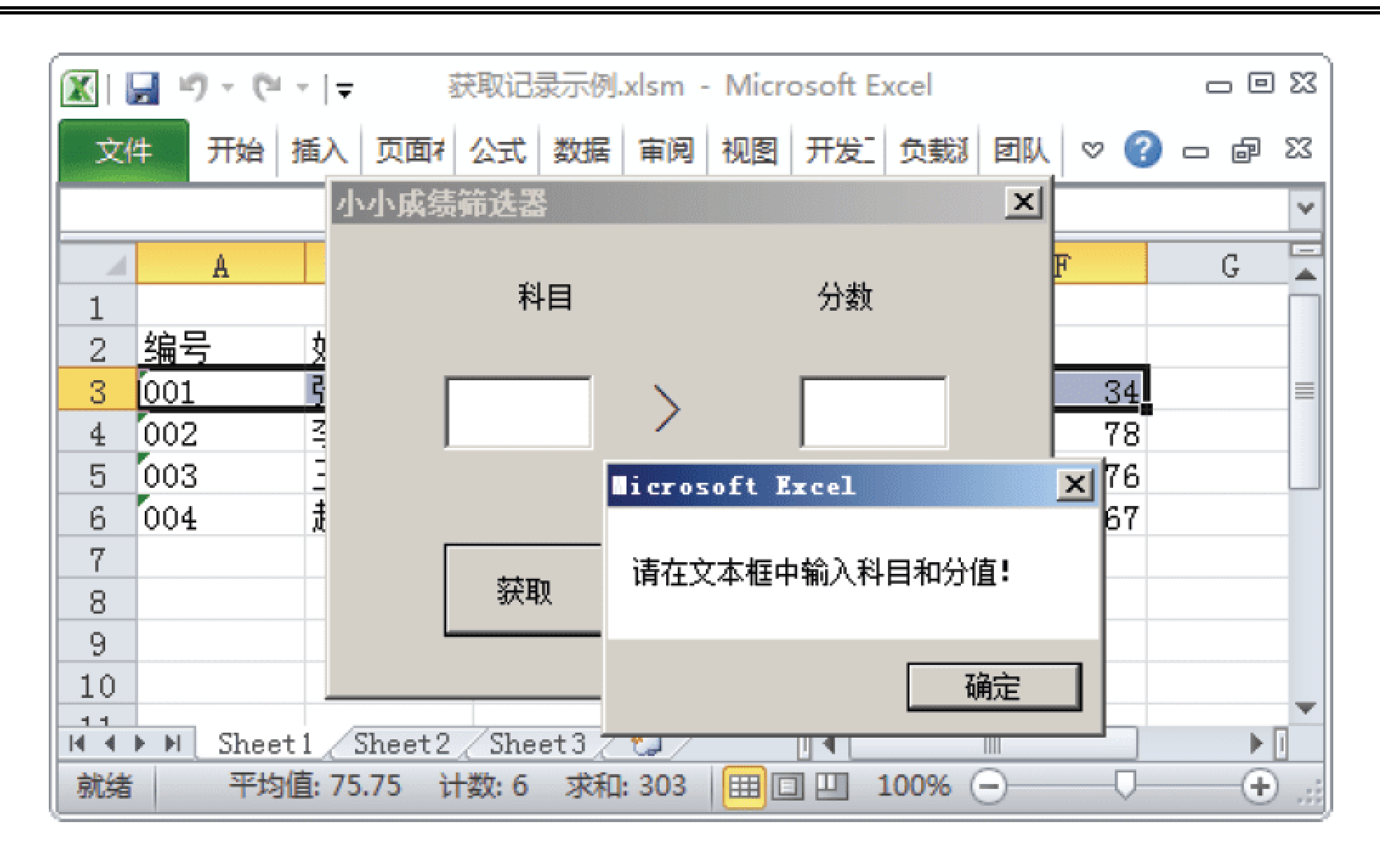


图 19.10　输入为空时给出提示

（3）为“清除”按钮添加 Click 事件代码。当单击该按钮时，2 个文本框被清空，第一个文本框获得焦点，等待用户继续输入，如图 19.11 所示。“清除”按钮的事件代码如下所示：

```
Private Sub CommandButton2 Click()
    TextBox1.Text = ""                      '清空“科目”文本框
    TextBox2.Text = ""                      '清空“分数”文本框
    TextBox1.SetFocus                       '“科目”文本框获得焦点
End Sub
```

19.2.4　在数据库中添加和删除记录

记录集对象 Recordset 主要用于完成数据的增加、删除和修改等操作。如果需要向记录集中添加一个记录，可以使用 AddNew 方法来实现，其语法格式为：

```
记录集.AddNew FieldList,Value
```

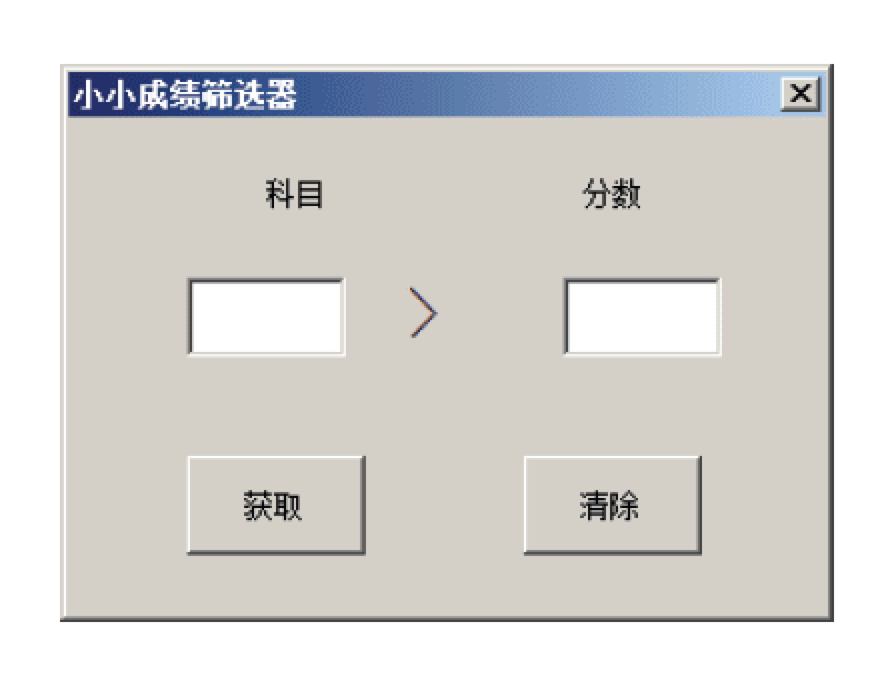


图 19.11　单击“清除”按钮后的效果

参数说明：

- ❑ FieldList 参数表示新记录中单个或一组字段的名称，也可表示在序列中的位置。
- ❑ Value 参数为新记录中字段的值。如果 Fields 是数字，则 Value 也必须是具有相同成员数的数字组，否则就会发生错误。字段名称的次序也必须与每个数组中字段值的次序相匹配。

如果需要将当前的记录和记录组删除，可以使用 Delete 方法来完成。其语法格式为：

```
记录集.Delete AffectRecords
```

这里，AffectRecord 决定 Delete 方法影响的记录数目。当该参数设置为 adAffectCurrent 时，仅删除当前记录，这个值为默认值。如果参数值为 adAffectGroup 时，删除满足当前 Filter 属性的记录。

如果语句完成了当前记录的修改或添加了新记录，需要对数据库进行更新（即保存），

可以使用 Update 方法来实现，该方法的语法结构为：

```
记录集.Update
```

注意：如果需要取消对数据库的修改，可以使用 CancelUpdate 方法来实现。

VBA 在访问一个记录集时，是按照记录的顺序来进行的，一次只能访问一个记录。正在访问的记录称为当前记录或活动记录。在打开任何一个记录集时，第一条记录都是当前记录。为了使其他记录成为当前记录，可以通过移动指示当前记录的记录指针的方法来实现。如，使用 MoveNext 方法或 MovePrevious 方法可以向前移一条记录或向后移一条记录。使用 MoveFirst 方法和 MoveLast 方法可以将记录指针移到记录集的第一条和记录集的最后一条。

如果需要将记录指针移动到记录集的指定位置，可以使用 Move 方法来实现，该方法的语法结构如下所示：

```
Move（NumRecords,Start）
```

参数说明：

- NumRecords 为数值表达式，表示相对于当前记录移动多少条记录，正数向后移，负数向前移。
- Start 指明移动的起始位置，该参数可取 3 个。当其为 adBookmarkCurrent 时，表示从当前记录开始，其为默认值；当其值为 adBookmarkFirst 时，表示从第一条记录开始；当其值为 adBookmarkLast 时，表示从最后一条开始。

在移动记录指针时，如果记录集在顶部时，执行 MovePrevious 方法，就会发生错误。同样，到记录集的底部时，使用 MoveNext 方法也会出现错误。这样需要判断记录指针是否指向了记录的顶部和底部，此时可以使用 BOF 和 EOF 属性来进行判断。

BOF 属性可指示当前记录位置为 Recordset 对象的第一个记录。如果当前记录位置位于第一个记录之前，其值为 True。如果当前记录就是第一个记录或位于其后，其值为 False。

EOF 属性指示当前记录位置为 Recordset 对象的最后一个记录。如果当前记录位于最后一个记录之后，其值为 True。如果当前记录为最后一个记录或位于其之前，则其值为 False。

下面通过一个实例来介绍使用 ADO 的 AddNew 方法来为数据库添加记录。程序运行时，单击窗体中的“添加”记录按钮，程序会依次给出输入对话框，要求输入新添加项目的各个字段的值，完成输入后该项目添加到数据库文件中。单击“删除记录”按钮将删除当前记录。本节实例的数据库使用前面创建的“成绩表.accdb”文件。

（1）打开 Visual Basic 编辑器，添加一个用户窗体。在用户窗体中放置 2 个“标签”控件和 2 个“命令按钮”控件，分别设置控件的属性。这里分别设置控件的 Caption 属性，并将第 2 个“标签”控件的文字设置为加粗显示。调整控件的大小和位置。完成设置后窗体控件的布局如图 19.12 所示。

（2）打开用户窗体的“代码”窗口，为窗体添加 Initialize 事件代码。该事件代码使窗体初始化时显示数据库中记录条数，如图 19.13 所示。

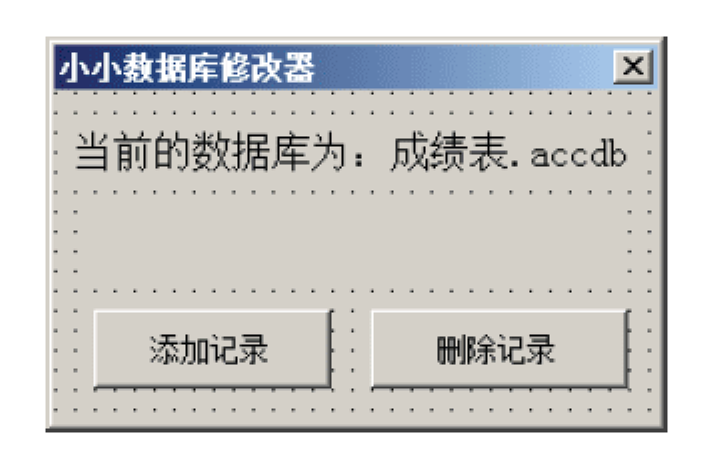



图 19.12　窗体中控件的布局

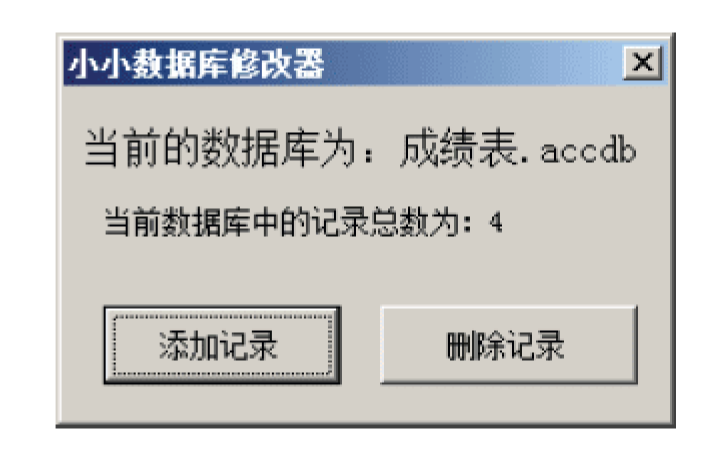


图 19.13　显示数据库记录总数

具体的窗体初始化事件代码如下所示：

```
01  Private Sub UserForm_Initialize()
02      Dim cn As ADODB.Connection                    '声明连接对象
03      Dim rs As ADODB.Recordset                     '声明记录集对象
04      Set cn = New ADODB.Connection                 '创建连接对象变量
05      Set rs = New ADODB.Recordset                  '创建记录集对象变量
06      With cn
07          .Provider = "microsoft.ace.oledb.12.0"    '指明 OLEDB 提供者
08          .Open ThisWorkbook.Path & "\成绩表.accdb"  '打开指定数据库
09      End With
10      rs.Open "成绩表1", cn, 1, 3               '打开数据库的“成绩表”表单
11      rs.MoveLast                               '指针移到记录最后
12      Label2.Caption = "当前数据库中的记录总数为：" _
13       & rs.RecordCount                         '显示记录总数
14      rs.Close                                  '关闭记录集对象
15      Set rs = Nothing
16      cn.Close                                  '关闭连接对象
17  End Sub
```

提示： 本段代码第 06~08 行是打开数据库操作。第 10 行代码在打开数据表时，Open 方法带上了参数，其中参数“1”表示游标类型为键集类指针，这能够保证记录指针在记录中任意移动。而参数“3”表示对打开的数据表具有读写权限。这里，如果不指定指针类型，运行第 11 行代码移动指针时，程序将报错。如果不开放读写权限，将无法添加记录。在第 12 和 13 行代码中，Recordset 的 RecordCount 属性返回 Recordset 对象中的当前记录数目。

（3）为“添加记录”按钮添加 Click 事件代码。“添加记录”按钮的鼠标单击事件代码如下所示：

```
Private Sub CommandButton1 Click()
    Dim cn As ADODB.Connection                    '声明连接对象
    Dim rs As ADODB.Recordset                     '声明记录集对象
    Set cn = New ADODB.Connection                 '创建连接对象变量
    Set rs = New ADODB.Recordset                  '创建记录集对象变量
    With cn
        .Provider = "microsoft.ace.oledb.12.0"    '指明 OLEDB 提供者
        .Open ThisWorkbook.Path & "\成绩表.accdb"'打开指定数据库
    End With
    rs.Open "成绩表1", cn, 1, 3                   '打开数据库的"成绩表"表单
    rs.MoveLast                                   '指针移到记录最后
    a = InputBox("请输入姓名：")                  '接收输入的姓名
    rs.AddNew                                     '添加新记录
```

```
        rs.Fields("姓名").Value = a                    '将姓名添加到记录中
        b = InputBox("请输入语文分数: ")                '接收输入的语文分数
        rs.Fields("语文").Value = b                    '将语文分数添加到记录中
        c = InputBox("请输入数学分数: ")                '接收输入的数学分数
        rs.Fields("数学").Value = c                    '将数学分数添加到记录中
        d = InputBox("请输入外语分数: ")                '接收输入的外语分数
        rs.Fields("外语").Value = d                    '将外语分数添加到记录中
        e = InputBox("请输入科学分数: ")                '接收输入的科学分数
        rs.Fields("科学").Value = e                    '将科学分数添加到记录中
        rs.Update                                      '更新数据库
        Label2.Caption = "当前数据库中的记录总数为: " _
         & rs.RecordCount                              '显示记录总数
        rs.Close                                       '关闭记录集对象
        Set rs = Nothing
        cn.Close                                       '关闭连接对象
        Set cn = Nothing
    End Sub
```

提示：在打开数据表后，使用 Inputbox 函数来获取用户的输入，在第一次输入姓名后，使用 Add 方法创建一个新记录，这个新记录就是数据表中的一个空行。使用 rs.Fields("姓名").Value = a 语句向指定的字段写入内容。在完成所有字段输入后，使用 Update 方法更新数据库。

程序运行时如果单击“添加记录”按钮，将给出提示对话框，要求输入各个字段的值，如图 19.14 所示。完成所有输入后，窗体显示当前记录总数，如图 19.15 所示。



图 19.14　输入字段的值

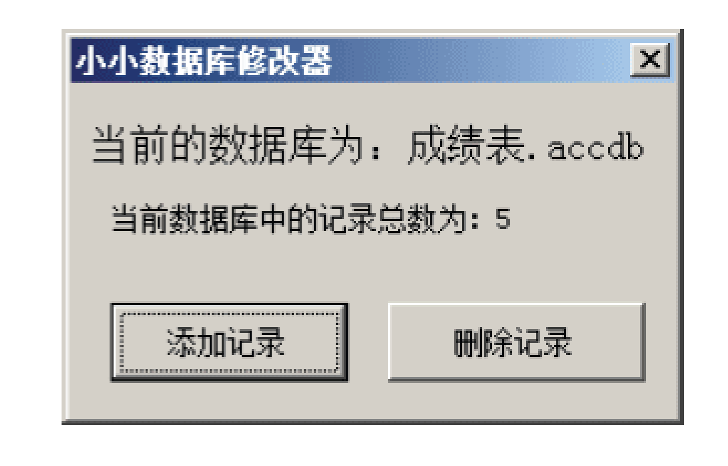


图 19.15　显示新的记录数

数据被添加到数据库中，如图 19.16 所示。

（4）为“删除记录”按钮添加 Click 事件代码。这里，在打开指定的表单后，将指针移到表单最后一个记录，直接使用 Delete 方法删除即可。删除记录后的窗体显示效果如图 19.17 所示。



图 19.16　数据添加到数据表的最后一行

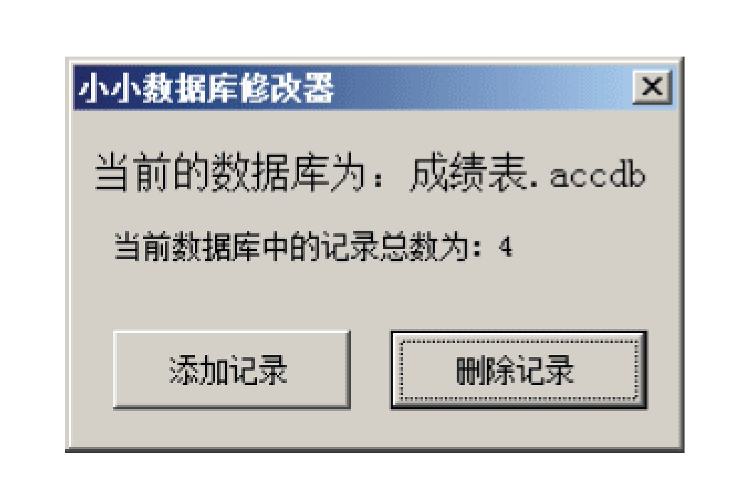


图 19.17　删除记录后显示新的记录条数

“删除记录”按钮的鼠标单击事件代码如下所示：

```
Private Sub CommandButton2_Click()
    Dim cn As ADODB.Connection                       '声明连接对象
    Dim rs As ADODB.Recordset                        '声明记录集对象
    Set cn = New ADODB.Connection                    '创建连接对象变量
    Set rs = New ADODB.Recordset                     '创建记录集对象变量
    With cn
        .Provider = "microsoft.ace.oledb.12.0"       '指明 OLEDB 提供者
        .Open ThisWorkbook.Path & "\成绩表.accdb"'打开指定数据库
    End With
    rs.Open "成绩表 1", cn, 1, 3                      '打开数据库的"成绩表"表单
    rs.MoveLast                                      '指针移到记录最后
    rs.Delete                                        '更新记录
    Label2.Caption = "当前数据库中的记录总数为：" _
     & rs.RecordCount                                '显示记录总数
    rs.Close                                         '关闭记录集对象
    Set rs = Nothing
    cn.Close                                         '关闭连接对象
    Set cn = Nothing
End Sub
```

提示：在程序中，rs.MoveLast 将记录指针移到记录最后一条，rs.Delete 语句将删除掉记录指针所指明的记录。

19.3　查询数据库中的数据

SQL 语言是 Structured Query Language（即结构化查询语言）的简称，这是一种标准的数据库语言，在任何关系型数据库管理系统中都可以使用。在使用 Excel 2010 VBA 进行数据管理时，经常需要从记录集中获取满足用户需要的数据，使用 SQL 语言是实现这种需要最为便捷的方法。SQL 语言并不仅仅是用于数据查询，还可以对记录进行更新、删除和添加等各种操作。

在数据库中，选取需要的记录是 SQL 语言最基本的功能。使用 Select 语句，可以从数据库的一个或多个表中选择一条记录的所有字段，也可以只选择记录中的部分字段。SELECT 语句的基本语法如下所示：

```
SELECT [TOP n]字段列表
FROM 表
WHERE 筛选条件
```

注意：在字段列表中如果指定多个字段名，在字段名之间应该用“,”分隔。如果需要指定表中所有的字段，可以用“*”来表示。如果只需要选择前 n 个数据，可以使用可选参数 Top n，n 为正整数。

下面通过介绍 SELECT 语句的常见用法。当需要查询“商品”表单中的所有数据时，可以使用下面语句来实现：

```
SELECT *
FROM 商品
```

如果需要查询“商品”表单中的“品名”、“单价”和“产地”字段，可以使用下面的语句：

```
SELECT 品名,单价,产地
FORM 商品
```

如果需要查询“商品”表单中的头 5 件商品的品名，可以使用下面的语句：

```
SELECT TOP5 品名
FROM 商品
```

如果需要在“商品”表单中查询产地为武汉的商品的品名，可以使用下面的语句：

```
SELECT 品名
FROM 商品
WHERE 产地="武汉"
```

提示： 在进行条件查询时，可以使用关系运算符>、<、>=、=、<>或 like 等。在进行条件查询时，可以使用通配符“*”和“%”，它们通配任意数量的任意字符。也可以使用“？”和“_”，它们表示通配一个字符。

下面通过一个实例来介绍使用 SQL 实现数据库查询的方法。本实例操作对象是一个名为“成绩管理”的 Access 数据库。在 VBA 中创建用户窗体，在用户窗体的“班级”列表框中选择班级后，可以在“选择学生”列表框中显示属于该班的学生名单。选择名单中的学生后，将能够在窗体的文本框中逐条显示该学生的各科成绩、考号以及姓名。本实例的详细制作过程介绍如下：

（1）使用 Access 2010 创建一个数据库，这个数据库是一个学生考试成绩表，结构如图 19.18 所示。



图 19.18　Access 数据库结构

（2）启动 Excel 2010，打开 Visual Basic 编辑器，按照前面介绍的方法完成 ADO 应用设置。添加一个用户窗体，在窗体中添加控件，这里添加了“框架”控件、“文本框”控件、“标签”控件和“命令按钮”控件。设置控件属性，这里将窗体中所有的文本框的“名称”属性设置得与数据库中对应的字段相同，如图 19.19 所示。

（3）添加窗体的 Initialize 事件代码。窗体初始化事件代码主要实现打开数据库，将数据库中的“班级”列表中的班级添加到列表框中，如图 19.20 所示。

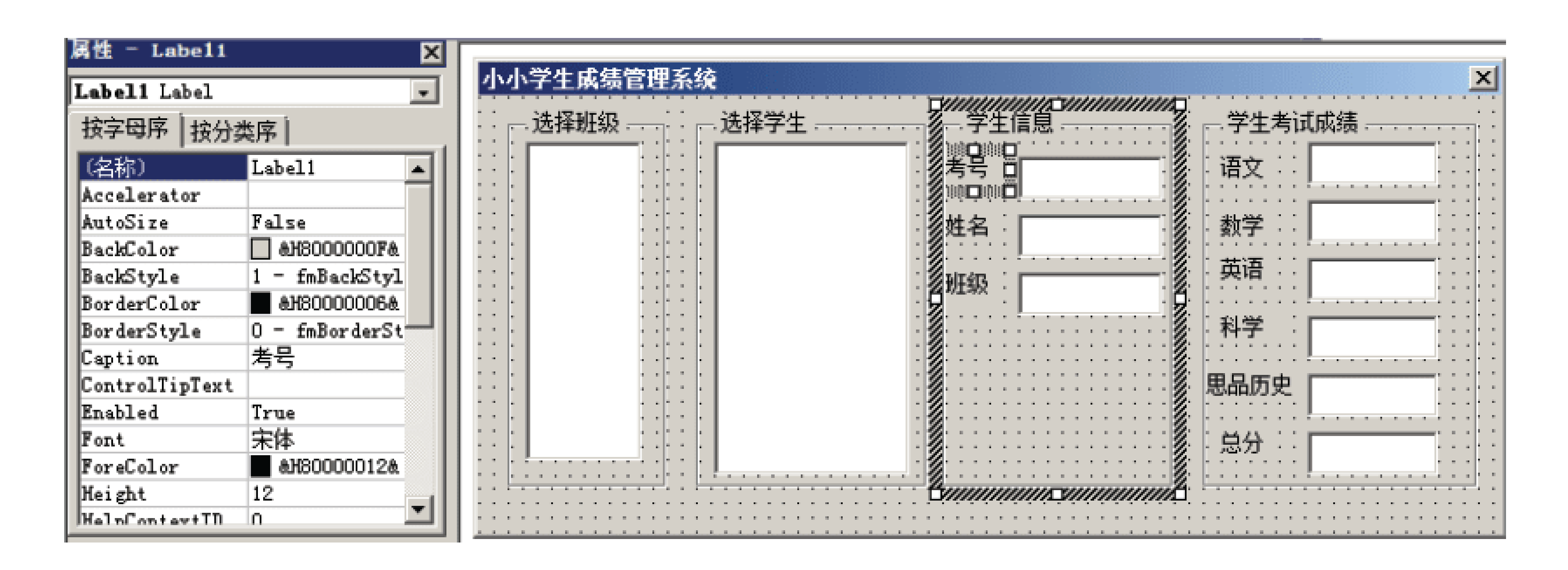


图 19.19　设置文本框属性



图 19.20　窗体初始化效果

窗体初始化事件的事件代码如下所示：

```
Dim cn As ADODB.Connection                                 '声明连接对象
Dim mydata As String, mytable As String                    '声明变量
Dim myArray As Variant                                     '声明数组变量
Private Sub UserForm_Initialize()
    Dim SQL As String                                      '声明过程变量
    Dim i As Integer
    mydata = ThisWorkbook.Path & "\成绩管理.accdb"    '指定数据库
    mytable = "成绩表"                                     '指定数据表
    myArray = Array("考号", "姓名", "班级", "语文", _
      "数学", "外语", "科学", "思品历史", "总分")          '创建数组
     Set cn = New ADODB.Connection                         '创建连接对象变量
    With cn
        .Provider = "microsoft.ace.oledb.12.0"             '指定数据提供者
        .Open mydata                                       '打开数据库
    End With
    SQL = "select distinct 班级 from " & mytable'创建 SQL 语句
    Set rs = New ADODB.Recordset                           '创建记录集对象变量
    rs.Open SQL, cn, adOpenKeyset, adLockOptimistic'创建查询结果记录集
    With ListBox1
        For i = 1 To rs.RecordCount                        '遍历所有班级名称记录
            .AddItem rs.Fields("班级")                     '将班级名称添加到列表中
            rs.MoveNext                                    '移动记录指针
        Next
    End With
End Sub
```

提示：程序的第 09 行代码将数据库的字段名放置于一个数组中，这样做有利于提高程序代码编写效率，也方便对字段名的引用和程序的修改。程序的第 16 行代码创建 SQL 命令，该命令的作用是查询数据表不同的班级名称。程序的第 16 行代码创建查询结果的记录集，读者请注意 SQL 语句在这里的使用方法。在第 20~25 行代码中，使用 For...Next 循环结构来遍历每个名称记录，将班级名称添加到“列表框”控件的列表中。

（4）下面为“选择班级”、“列表框”控件添加鼠标单击事件。当选择列表框中班级选项时，在“选择学生”列表框中将列出数据库中这个班的所有学生，如图 19.21 所示。



图 19.21　选择列表中选项后的效果

控件的鼠标单击事件代码如下所示：

```
01  Private Sub ListBox1 Click()
02      Dim SQL As String                          '声明变量
03      Dim i As Integer
04      Dim rs As ADODB.Recordset                  '声明记录集变量
05      SQL = "select 姓名 from " & mytable _
06      & " where 班级='" & ListBox1.Value & "'order by 考号"
                                                   '查询班级的所有学生姓名
07      Set rs = New ADODB.Recordset               '创建记录集变量
08      rs.Open SQL, cn, adOpenKeyset, adLockOptimistic'获得查询结果的记录集
09      With ListBox2
10          .Clear                                 '清除原有项目
11          For i = 1 To rs.RecordCount            '遍历所有记录
12              .AddItem rs.Fields("姓名")         '将设置为列表框选项
13              rs.MoveNext                        '指针下移
14          Next
15      End With
16  End Sub
```

提示：在第 05～06 行代码中，select 姓名 from " & mytable & " where 班级='" & ListBox1.Value 表示选择查询数据库中“姓名”字段的记录，该记录应该满足的条件是“班级”为第一个“列表框”控件选项，而"'order by 考号"表示根据考号来排序。在第 11～15 行代码中，使用 For...Next 循环结构来遍历所有的记录，并将记录依次设置为“列表框”控件的选项。

（5）下面为“选择学生”列表框控件添加鼠标单击事件代码。这段代码的作用是将选

择的学生的成绩以及有关信息在右侧的文本框中列出来，如图 19.22 所示。

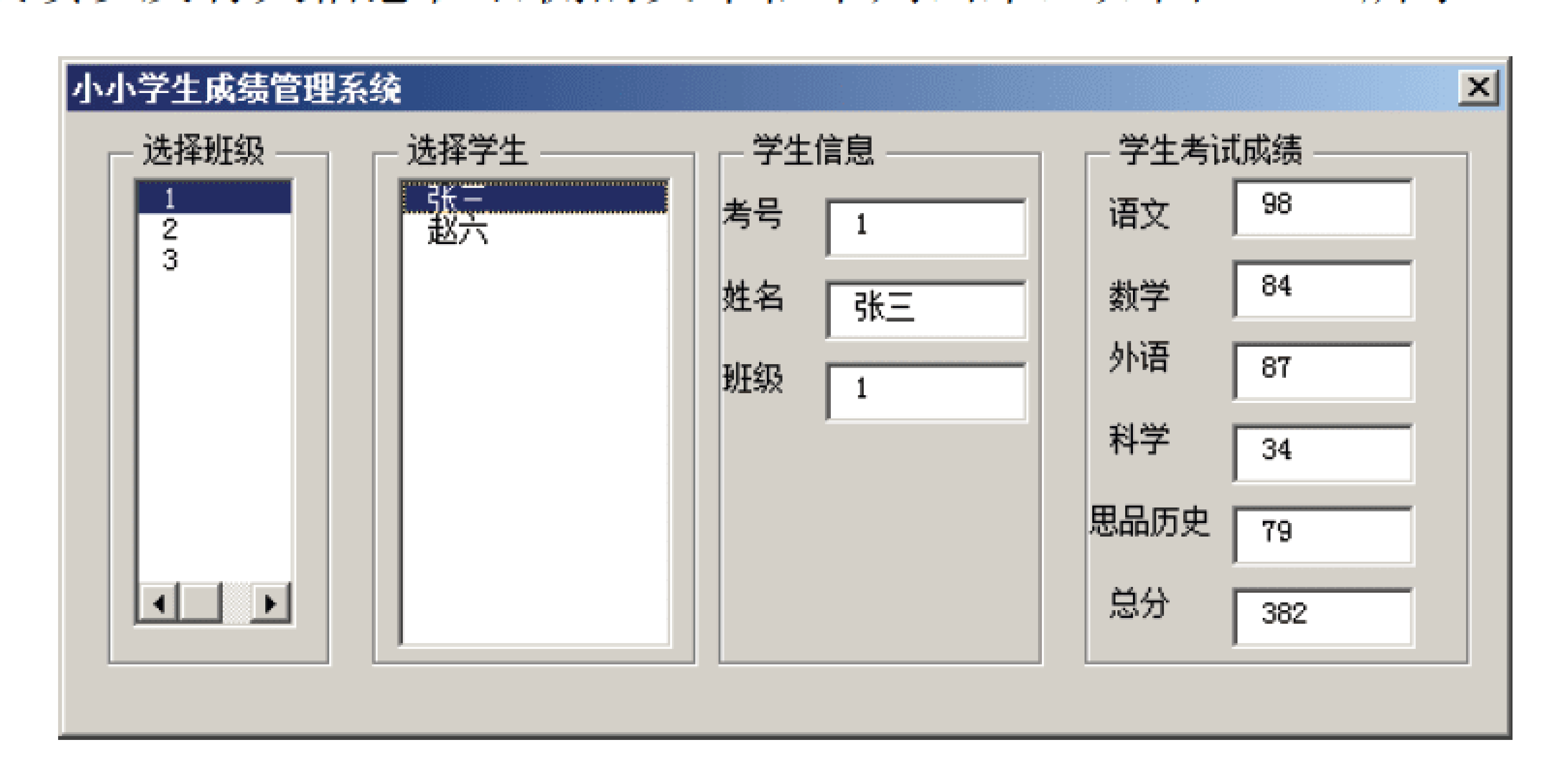


图 19.22　选择学生后的显示效果

事件的鼠标单击事件代码如下所示：

```
01  Private Sub ListBox2_Click()
02      Dim SQL As String
03      Dim i As Integer
04      Dim rs As ADODB.Recordset                          '声明记录集变量
05      SQL = "select * from" & mytable & " where 班级='" & ListBox1.Value &"'"_
06        & " and 姓名='" & ListBox2.Value & " ' order by 考号"
                                                           '查询学生的考试成绩
07      Set rs = New ADODB.Recordset                       '创建记录集变量
08      rs.Open SQL, cn, adOpenKeyset, adLockOptimistic'获得查询结果记录集
09      For i = 0 To UBound(myArray)                       '变量数组中的每一项
10          Me.Controls(myArray(i)).Value = rs.Fields(i)'文本框中显示学生的成绩
11      Next
12  End Sub
```

提示： 这里，第 05~06 行代码设置 SQL 命令，第 09~11 行代码使用 For...Next 循环结构将各个字段的记录写入对应的文本框。程序中使用 Controls 集合对象来访问窗体上的控件集合，myArray(i)指明控件对象，因为 myArray 数组中的元素是窗体中文本框控件的名称。这也就是本实例的第（1）步在“属性”对话框中需要更改“文本框”控件的名称的原因。

19.4　小　　结

本章介绍了数据库的基础知识、在 VBA 中使用 ADO 数据对象模型操作数据库的方法，同时还介绍了使用 SQL 语言查询数据库记录的方法。通过本章的学习，读者将能够使用 VBA 实现对数据库中数据的筛选、数据的添加和删除以及实现数据库与 Excel 工作表间的数据交流。

SQL 查询语言是操作数据库的基本语言，在进行数据库操作时，其经常与 ADO 数据对象模型配合使用。在 VBA 中进行数据库的操作时，一般的步骤是：使用 Connection 对象建立数据源，然后使用 Recordset 对象打开记录数据集，此时也可以通过执行 SQL 语句

来获得记录集对象，接着使用 Recordset 对象或 Field 对象来调用数据集中的数据，最后在完成操作后中断对象连接。

19.5　本 章 习 题

1．下面引用字段错误的是？（　　）

A．rs.fields("姓名 ID")　　　　B．rs！姓名 ID

C．rs.fields(0)　　　　D．rs.姓名 ID

2、下面对数据库的筛选语句中哪项是正确的？（　　）

A．rs.Filter="姓名"As"王鹏"　　　　B．rs.Filter="姓名"."王鹏"

C．rs.Filter="姓名"Like"王*"　　　　D．rs.Filter="姓名"_"王鹏"

3．下面程序的作用是什么？（　　）

```
Public Sub test()
    Dim mydata As String, mytable As String
    Dim cnn As ADODB.Connection
    Dim rs As ADODB.Recordset
    mydata = ThisWorkbook.Path & "\客户管理.mdb"     '指定数据库文件
    mytable = "客户信息"                             '指定要查询的数据表名称
    Set cnn = New ADODB.Connection
    With cnn
        .Provider = "microsoft.jet.oledb.4.0"
        .Open mydata
    End With
    Set rs = cnn.OpenSchema(adSchemaTables)
    Do Until rs.EOF
        If LCase(rs!table name) = LCase(mytable) Then
            MsgBox "数据表 < " & mytable & ">  存在！"
            GoTo tuich
        End If
        rs.MoveNext
    Loop
    MsgBox "数据表 " & mytable & " 不存在！"
tuich:
    rs.Close
    cnn.Close
    Set rs = Nothing
    Set cnn = Nothing
End Sub
```

A．在“客户管理”数据库中打开“客户信息”工作表

B．在“客户管理”数据库中查询是否存在字符“客户信息”

C．在“客户管理”数据库中查询“客户信息”字段

D．在“客户管理”数据库中查询是否存在“客户信息”数据工作表

4．使用 SQL 语言查询 19.3 节的“成绩管理”数据库，筛选出 1 班的数学和语文成绩均大于 98 分的学生，并且将这些学生的记录复制到 Excel 工作表中。

【提示】使用 SQL 语言能够方便地进行数据的查询，本练习是一个多条件的查询，满

足条件记录可以用Select语句来获得，"select * from " & mytable & " where (班级='1') and (数学>=98 and 语文>=98)"。获取对象后，使用 For...Next 循环结构将数据表的各个字段名，写入单元格，然后使用 CopyFromRecordset 方法复制记录即可。

5．在 19.2.3 节使用的“成绩表”数据库中添加一个名为“总分”的字段。

【提示】本例使用 SQL 语言中的 ALTER 语句，该语句能够实现对数据库结构的修改。该语句使用比较简单，只需要指定工作表和需要添加的字段名及其数据类型，引用该语句即可执行添加字段的操作。

第 3 篇　项目开发案例实战

⏭　第 20 章　教务管理系统

⏭　第 21 章　档案管理系统

第 20 章　教务管理系统

教务管理系统是将相关考试成绩录入到系统当中，提供给学生和教师一定的操作界面，实现对考试成绩的管理和查询，通过系统主要实现对不同权限的用户对应不同的操作，即学生只能查询自己的成绩，教师能依据相应的条件过虑相关的学生记录，同时提供管理员权限，实现对系统中成绩的查询。本章学习的内容如下：

- ❑ 掌握用户登录界面的设计方法；
- ❑ 掌握不同权限的功能设计方法；
- ❑ 掌握 Excel 2010 中数据高级查询方法；
- ❑ 掌握对工作表以及 VBA 程序的保护方法。

20.1　设 计 功 能

本章介绍一个教务管理系统的制作过程。是针对不同的用户需要赋予用户不同的权限级别，使用户具有不同的操作能力。本节将首先介绍实例的功能和制作思路。

20.1.1　功能简介

首先，本系统是一个教务管理系统，允许学生和老师对工作表中的学生成绩进行基于不同条件的查询。由于学生和老师对查询的要求不同，因此在程序中，为他们分别设置了不同的操作权限，以实现对操作者的分级管理。

系统用来管理不同级别的用户，使不同的用户在操作时，只能查看涉及其权限允许范围内的工作表和信息，并只能对工作表进行有限的规定操作。这样做的目的是为了有效地保证数据的安全，保护工作表和程序不被没有权限的用户随意更改。

本实例中，用户的权限级别分为 3 个等级，它们是学生、教师和管理员。在打开工作表时，程序给出登录界面，用户可以选择登录的类别。作为学生登录，权限最小，只能通过姓名对成绩进行查询，大多数工作表对这类用户都不可见。在登录窗口中选择登录身份为学生后，直接在登录窗口中输入姓名即可查询成绩，如图 20.1 所示。

如果以教师身份登录，教师拥有的权限高于学生，除了能够查看学生看到的一切信息，还能够通过高级查询功能对成绩进行更为复杂的查询，以便于对考试情况进行分析。对于涉及系统的重要的数据表和程序代码无权查看和修改。在登录窗口中选择以教师身份登录，输入姓名和密码，如图 20.2 所示。单击“登录”按钮后，可以打开“学生成绩调查表”，在调查表中输入需要查询的条件后，单击工作表中的“开始筛选”按钮，工作表中将会列

出查到的结果，如图 20.3 所示。

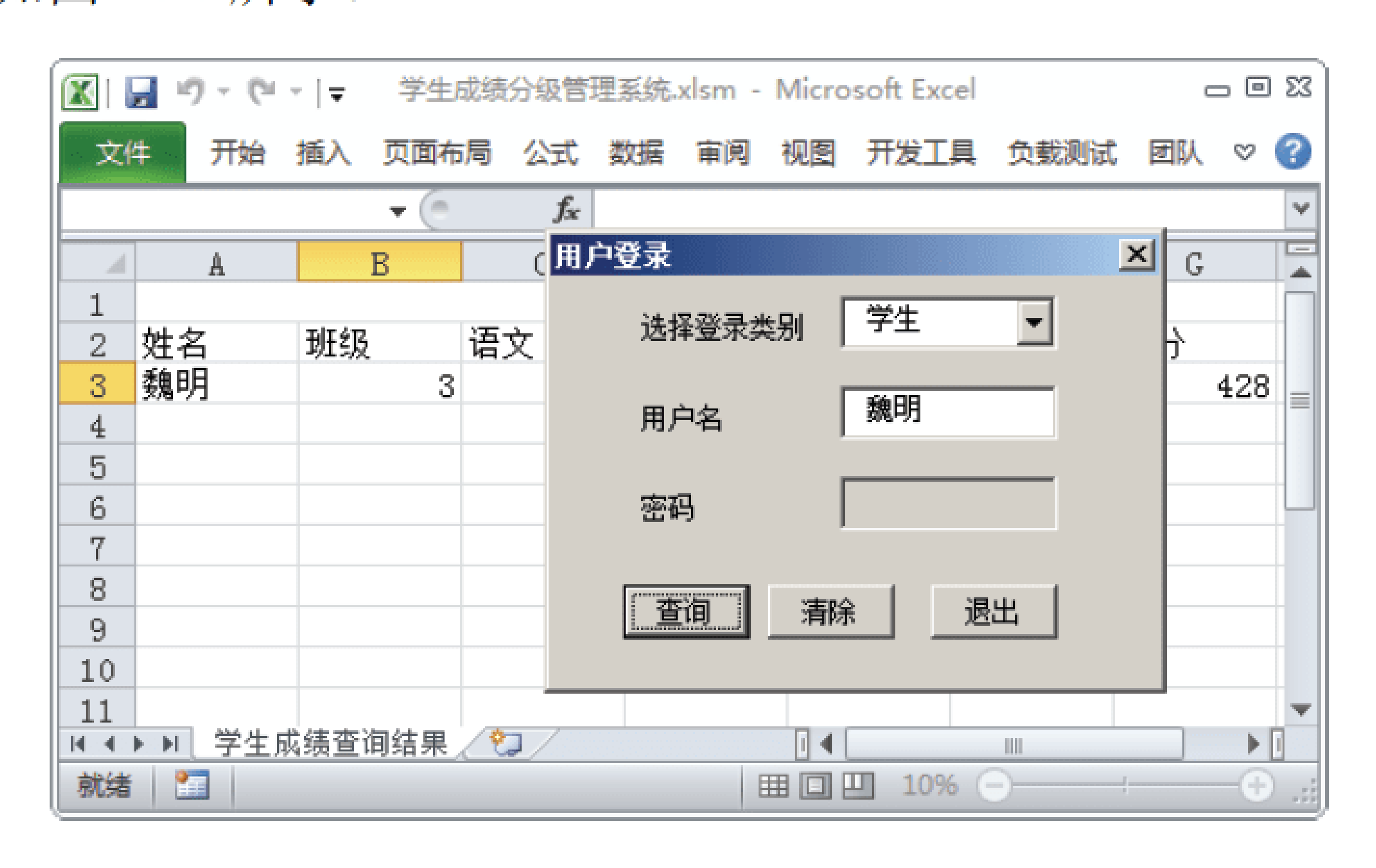


图 20.1　查询学生成绩

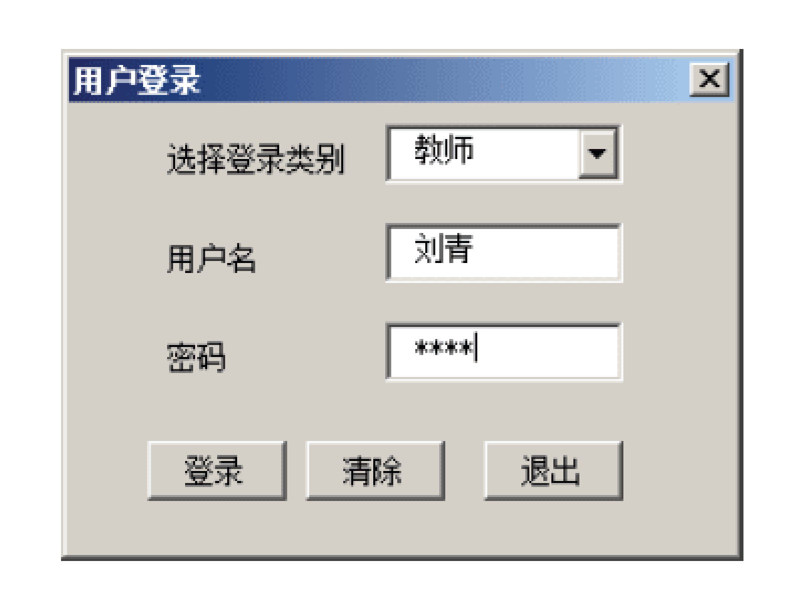


图 20.2　以教师身份登录



图 20.3　工作表中列出查询的结果

管理员是这个系统中最高权限的拥有者。能够查看和修改工作簿中所有的工作表，能够对程序进行修改，并可对可登录的教师进行管理。在登录窗口中选择以管理员身份登录，登录成功后，文档中所有工作表均可见，如图 20.4 所示。在 VBA 编辑器中，打开工程，需要输入管理员密码以保证只有管理员才能对程序进行修改。

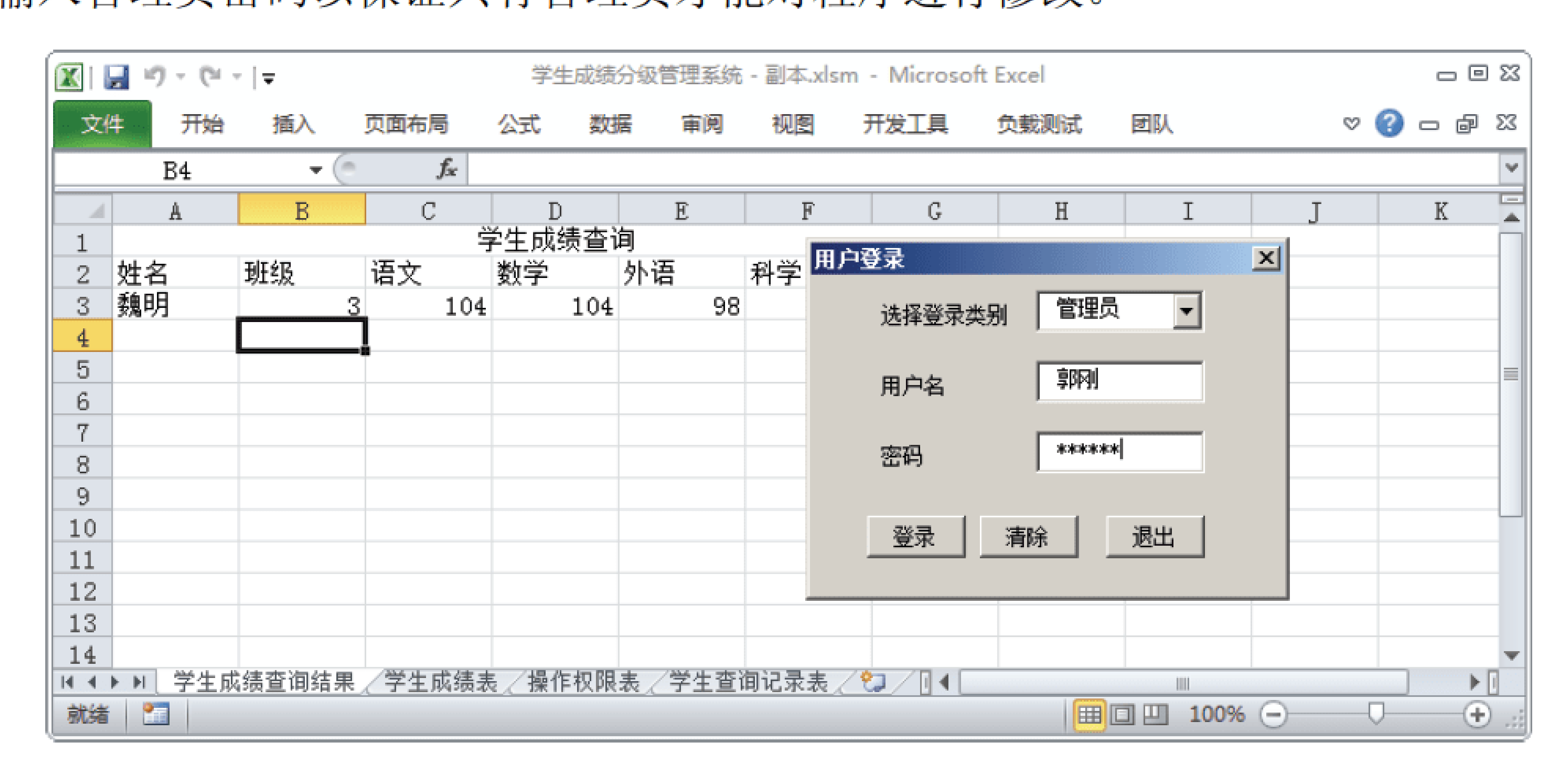


图 20.4　以管理员身份登录能查看所有工作表

20.1.2　设计思路

本实例的用户权限的选择，通过一个登录界面来实现。通过登录界面，操作者可以以不同的身份登录，登录后具有对工作表的不同操作能力。实例制作顺序是首先设计用户界面，然后实现学生成绩查询和教师成绩查询功能，最后实现管理员对系统的管理能力。本实例的制作采用模块化的制作方式，根据程序功能的需要，对本实例系统的模块进行划分，如图 20.5 所示。

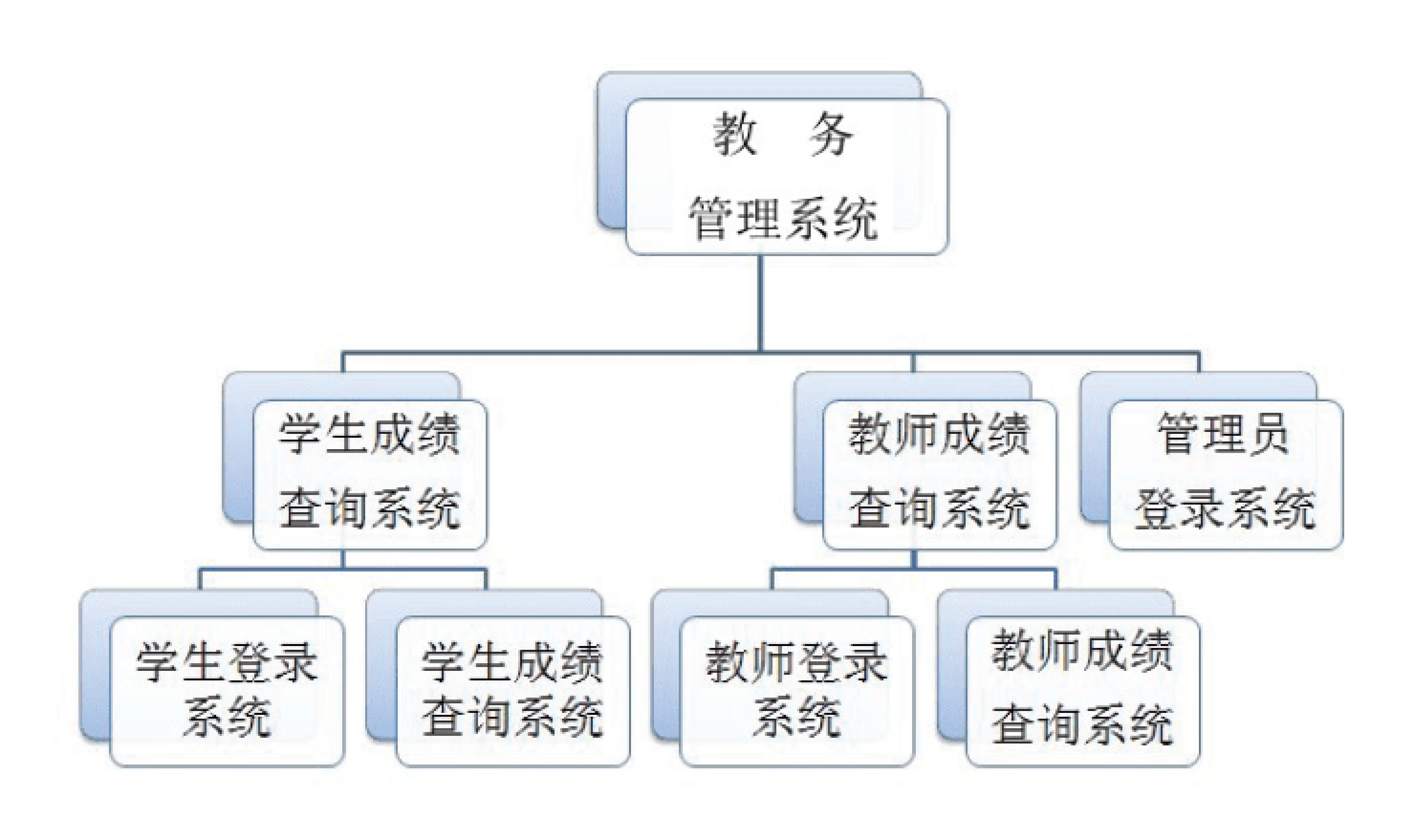


图 20.5　本实例的功能模块图

20.2　设计用户界面

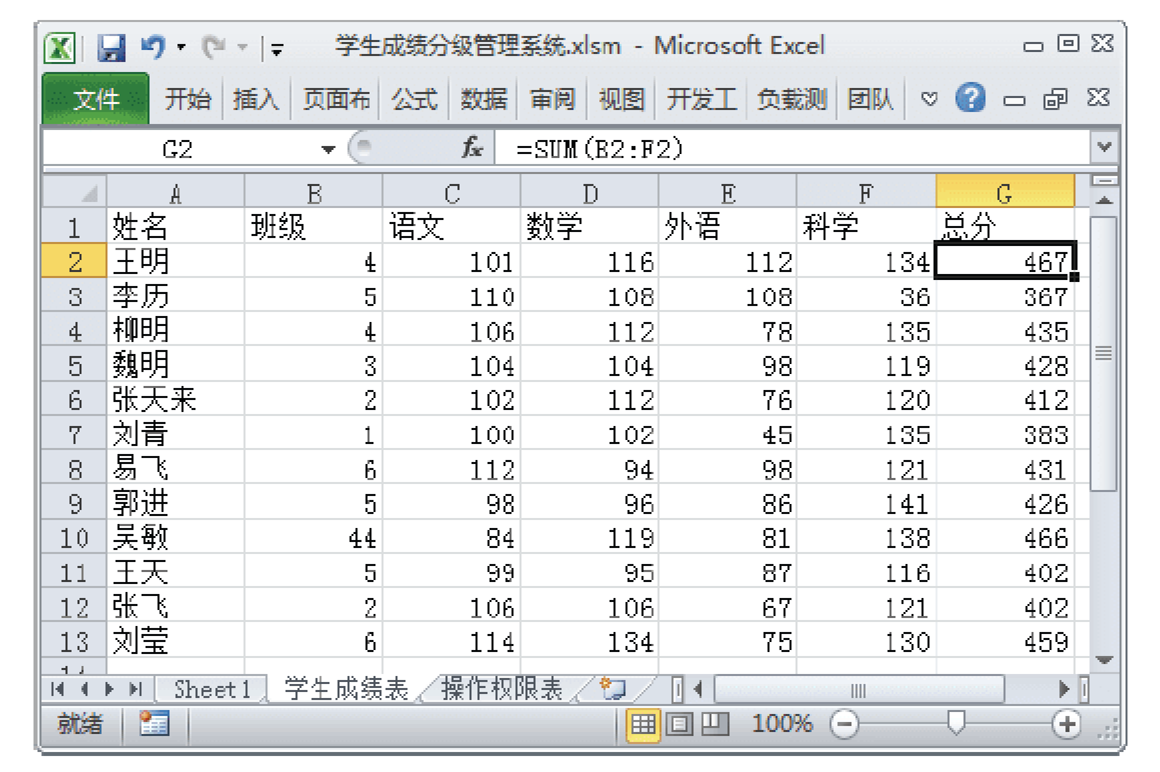
用户界面的设计包括 Excel 工作表的设计和用户操作界面的设计。在本例中，用户操作界面实际上就是一个用户分级登录窗体，使用该窗体来实现不同用户的不同操作权限的选择。

20.2.1　创建 Excel 2010 工作表

为实现登录用户的选择，首先需要创建 2 个工作表，它们分别是“学生成绩表”和“操作权限表”。

（1）启动 Excel 2010，新建工作簿。删除工作簿中多余的表格，只保留 3 个工作表，并将工作表“Sheet2”和“Sheet3”命名为“学生成绩表”和“操作权限表”。在“学生成绩表”中输入学生成绩信息，如图 20.6 所示。

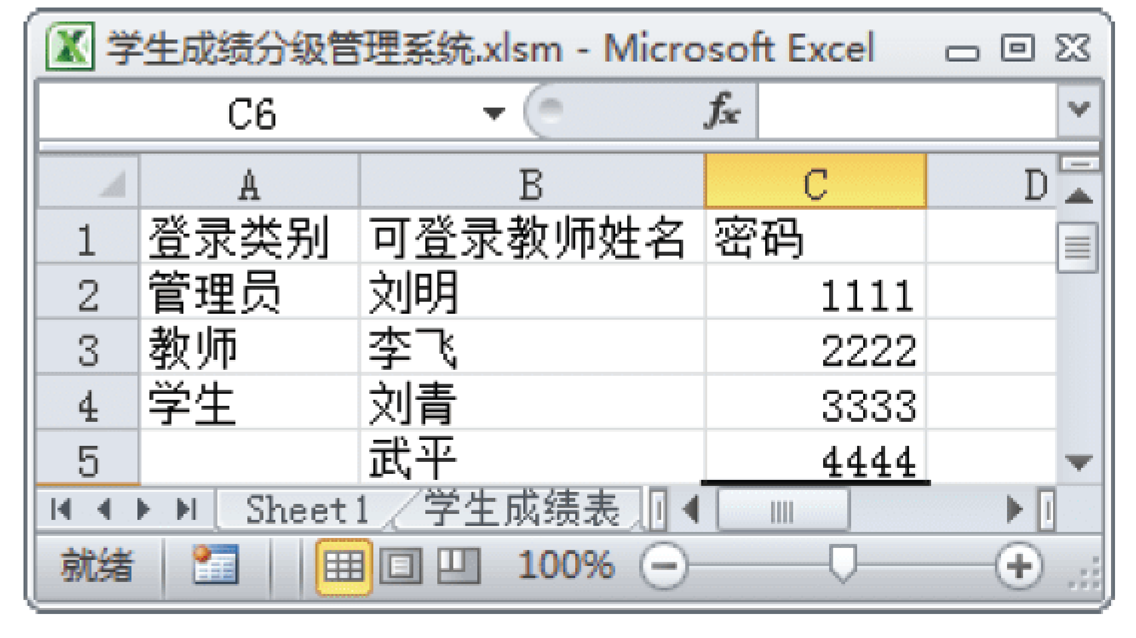
（2）打开“操作权限表”工作表，在工作表中输入有关信息，如图 20.7 所示。这个工作表的第 A 列为登录界面中“选择登录类别”下拉列表提供登录类别选项，B 列和 C 列提供了可以登录的教师名单和登录密码，这里一共设置了 4 位可登录的教师。



姓名	班级	语文	数学	外语	科学	总分
王明	4	101	116	112	134	467
李历	5	110	108	108	36	367
柳明	4	106	112	78	135	435
魏明	3	104	104	98	119	428
张天来	2	102	112	76	120	412
刘青	1	100	102	45	135	383
易飞	6	112	94	98	121	431
郭进	5	98	96	86	141	426
吴敏	44	84	119	81	138	466
王天	5	99	95	87	116	402
张飞	2	106	106	67	121	402
刘莹	6	114	134	75	130	459



图 20.6　输入学生信息



登录类别	可登录教师姓名	密码
管理员	刘明	1111
教师	李飞	2222
学生	刘青	3333
	武平	4444

图 20.7　“操作权限表”工作表

20.2.2　设计教务管理登录界面

程序运行时将显示一个登录窗体，用户通过登录窗体检验身份后根据不同身份进行操作。这里，设计登录窗体包括窗体控件的添加和设置以及窗体初始化程序的编写。

（1）打开 Visual Basic 编辑器。在工程资源管理器中添加一个用户窗体，向窗体中添加 3 个“标签”控件、1 个“组合框”控件、2 个“文本框”控件和 3 个“命令按钮”控件。分别设置用户窗体、“标签”控件和“命令按钮”控件的 Caption 属性，同时调整各个控件在窗体中的位置和大小。窗体设计完成后的控件布局，如图 20.8 所示。

（2）为了输入密码时密码不会显示，应该在“密码”文本框的“属性”对话框中将其 PasswordChar 属性设置为“*”，如图 20.9 所示。这样，输入密码时，密码字符将被“*”所代替。



图 20.8　窗体控件的布局

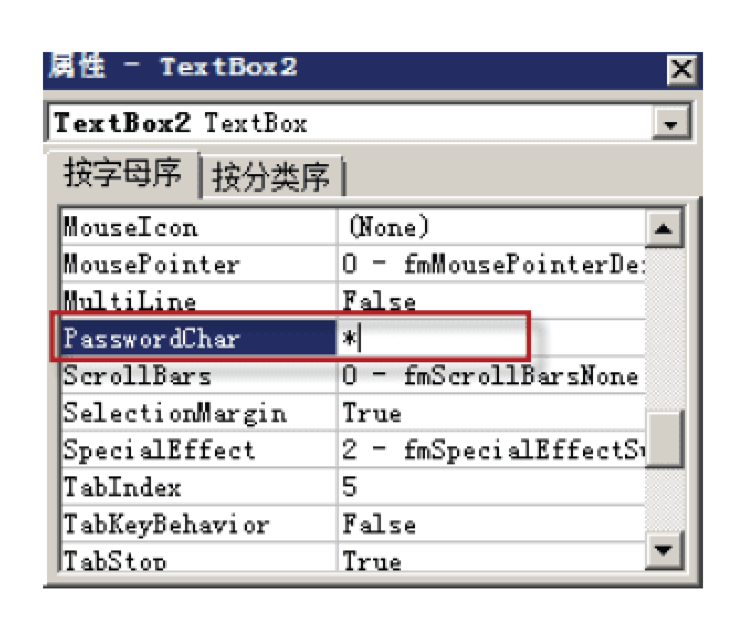


图 20.9　设置 PasswordChar 属性

提示： 窗体中的“组合框”控件用于选择登录人员的类别，“姓名”文本框用于输入登录者的姓名，而“密码”文本框用于输入登录密码。

（3）在用户窗体上双击打开窗体的“代码”窗口，为窗体添加 Initialize 事件代码。初始化过程首先为窗体中的“选择登录类别”文本框添加选项，选项来源于“操作权限表”工作表的第 1 列。在没有选择登录类别时，为了避免用户在文本框中输入内容，将窗体中的文本框设置为灰色不可用状态。窗体的初识化事件代码如下所示：

```
Private Sub UserForm_Initialize()
    Dim i As Integer, j As Integer
```

```
    TextBox1.Enabled = False                    '“姓名”文本框不可用
    TextBox1.BackColor = &H8000000F             '“姓名”文本框背景设置为灰色
    TextBox2.Enabled = False                    '“密码”文本框不可用
    TextBox2.BackColor = &H8000000F             '“密码”文本框背景设置为灰色
    j = Sheets("操作权限表").Range("A1048576"). _
    End(xlUp).Row                               '获得最后一个非空单元格行号
    For i = 2 To j                              '遍历所有单元格
        ComboBox1.AddItem Sheets("操作 _
        权限表").Cells(i, 1)                     '为控件添加选项
    Next
End Sub
```

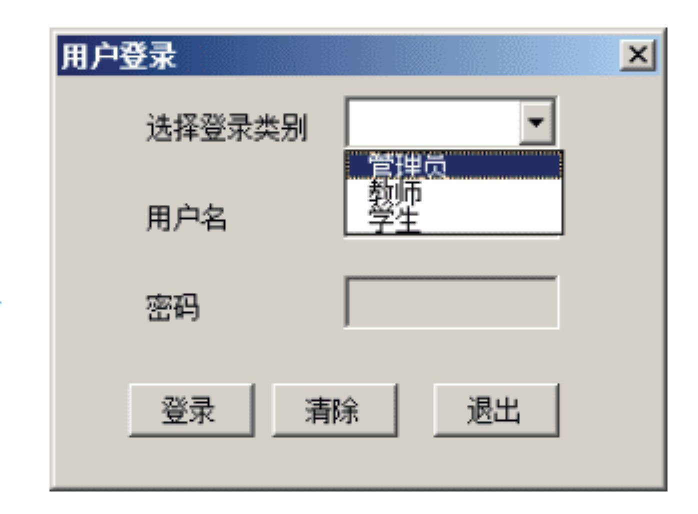


图 20.10　窗体初始化效果

提示：在这段代码中，将控件的 Enable 属性设置为 False，控件将不可用。控件的 BackColor 属性用于设置控件背景颜色，颜色值&H8000000F 表示灰色。在“操作权限表”的 A 列类别“教师”和“学生”间存在着空白单元格。

窗体初始化事件代码执行后的窗体效果，如图 20.10 所示。

20.3　学生查询分数

本实例允许以学生的身份来查询成绩。当登录类别选择“学生”时，在“姓名”对话框中输入学生姓名，查询的结果将写入单独的工作表中。同时，系统将记录学生登录查询的姓名和时间。

（1）为“组合框”控件添加 Change 事件代码。“组合框”控件的 Change 事件代码如下所示：

```
Private Sub ComboBox1_Change()
    If ComboBox1.Value = "学生" Then            '如果选择的是学生
        CommandButton1.Caption = "查询"         '“登录”按钮变为“查询”按钮
        TextBox1.Enabled = True                 '“姓名”文本框可用
        TextBox1.BackColor = &H80000005         '“姓名”文本框恢复为正常的白色
        TextBox2.Enabled = False                '“密码”文本框不可用
        TextBox2.BackColor = &H8000000F         '“密码”文本框变为灰色
        TextBox1.SetFocus                       '“姓名”文本框获得焦点
    End If
    If ComboBox1.Value = "管理员" Or ComboBox1 _
    .Value = "教师" Then                    '如果选择的是“管理员”或“教师”
        CommandButton1.Caption = "登录"         '恢复按钮名称为“登录”
        TextBox1.Enabled = True                 '“姓名”文本框可用
        TextBox2.Enabled = True                 '“登录”文本框可用
        TextBox1.BackColor = &H80000005         '文本框颜色恢复为正常的白色
        TextBox2.BackColor = &H80000005
        TextBox1.SetFocus                       '“姓名”文本框获得焦点
    End If
End Sub
```

这里，当类别选择“学生”时，“密码”文本框应该不可用。同时，学生查询功能单一，可以直接根据输入的姓名进行查询，此时“登录”按钮变为“查询”按钮，如图 20.11 所示。选择“教师”或“管理员”类别时，“姓名”和“密码”文本框同时可用，如图 20.12 所示。







图 20.11　选择“学生”时的登录窗口　　　图 20.12　选择“教师”时的登录窗口

提示： 当“组合框”控件的选项改变时，触发 Change 时间，程序将执行。事件代码主要是根据不同的选择来更改控件的外观，如改变“命令按钮”控件 CommandButton1 的 Caption 属性改变按钮显示的名称。这里，以控件的 Value 属性值作为判断登录类型的依据，为了避免学生以“教师”身份登录，教师和管理员一样，在登录时都需要输入密码。

（2）切换到 Excel 2010 操作界面，将“sheet1”工作表更名“学生成绩查询结果”。该工作表用于显示学生分数查询结果，如图 20.13 所示。

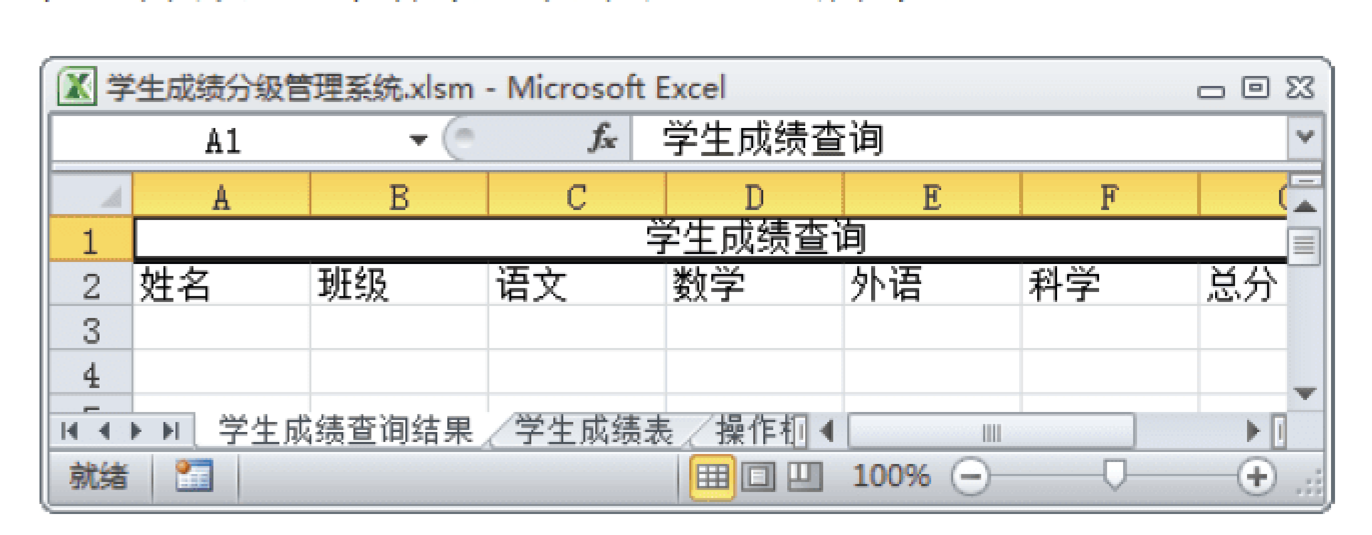
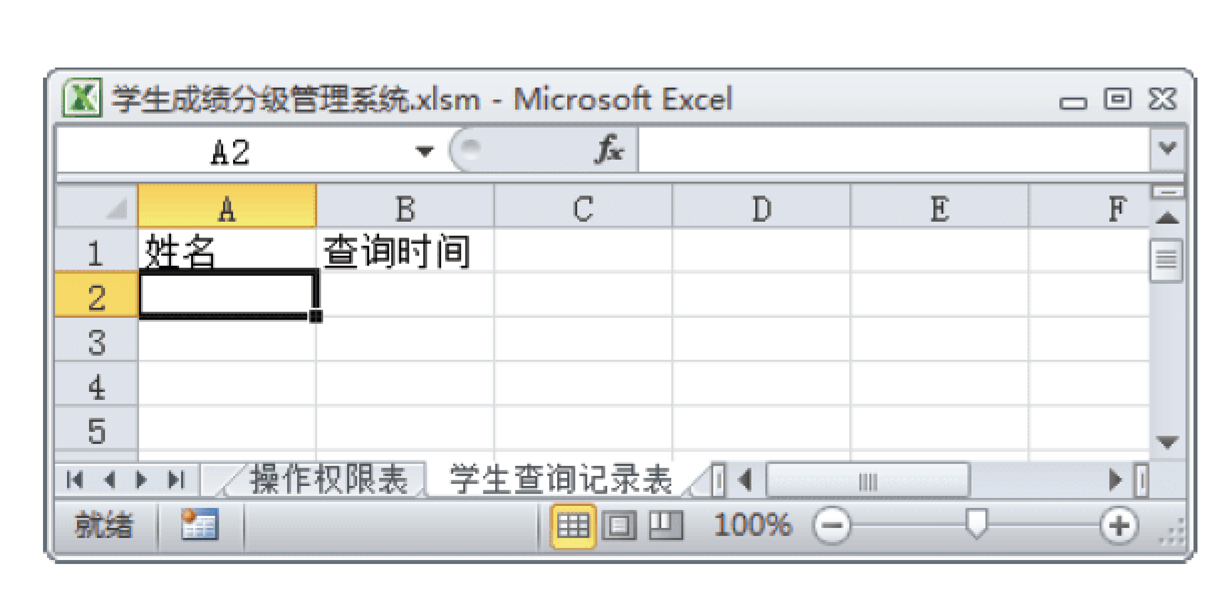


图 20.13　设计“学生成绩查询结果”工作表

（3）在工作簿中插入一个名为“学生查询记录表”的工作表，该工作表用于记录学生姓名和登录时间，如图 20.14 所示。选择工作表中 B 列的所有单元格，打开“设置单元格格式”对话框，设置该列单元格的格式，如图 20.15 所示。



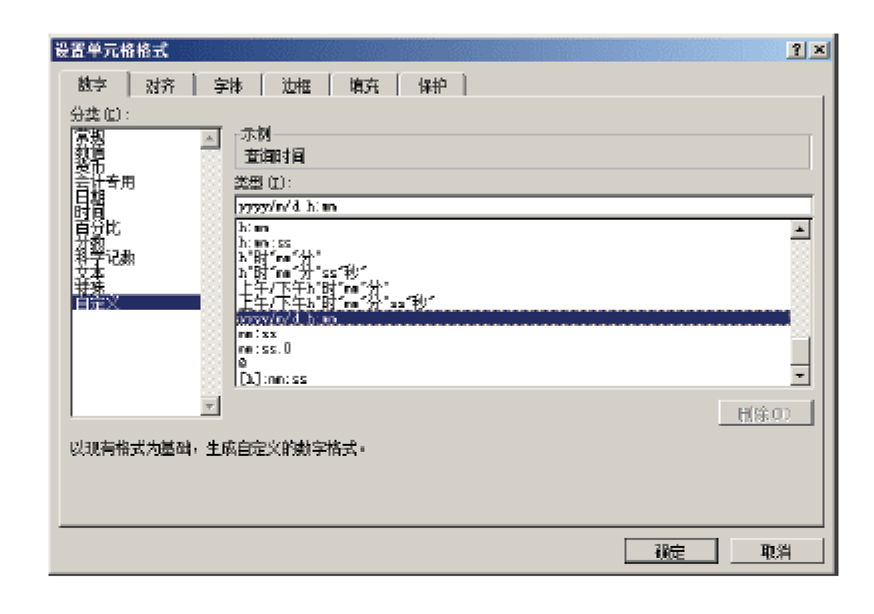


图 20.14　“学生查询记录表”工作表　　　图 20.15　设置单元格格式

（4）为“登录”按钮添加 Click 事件代码，实现学生成绩查询功能，程序流程如图 20.16 所示。

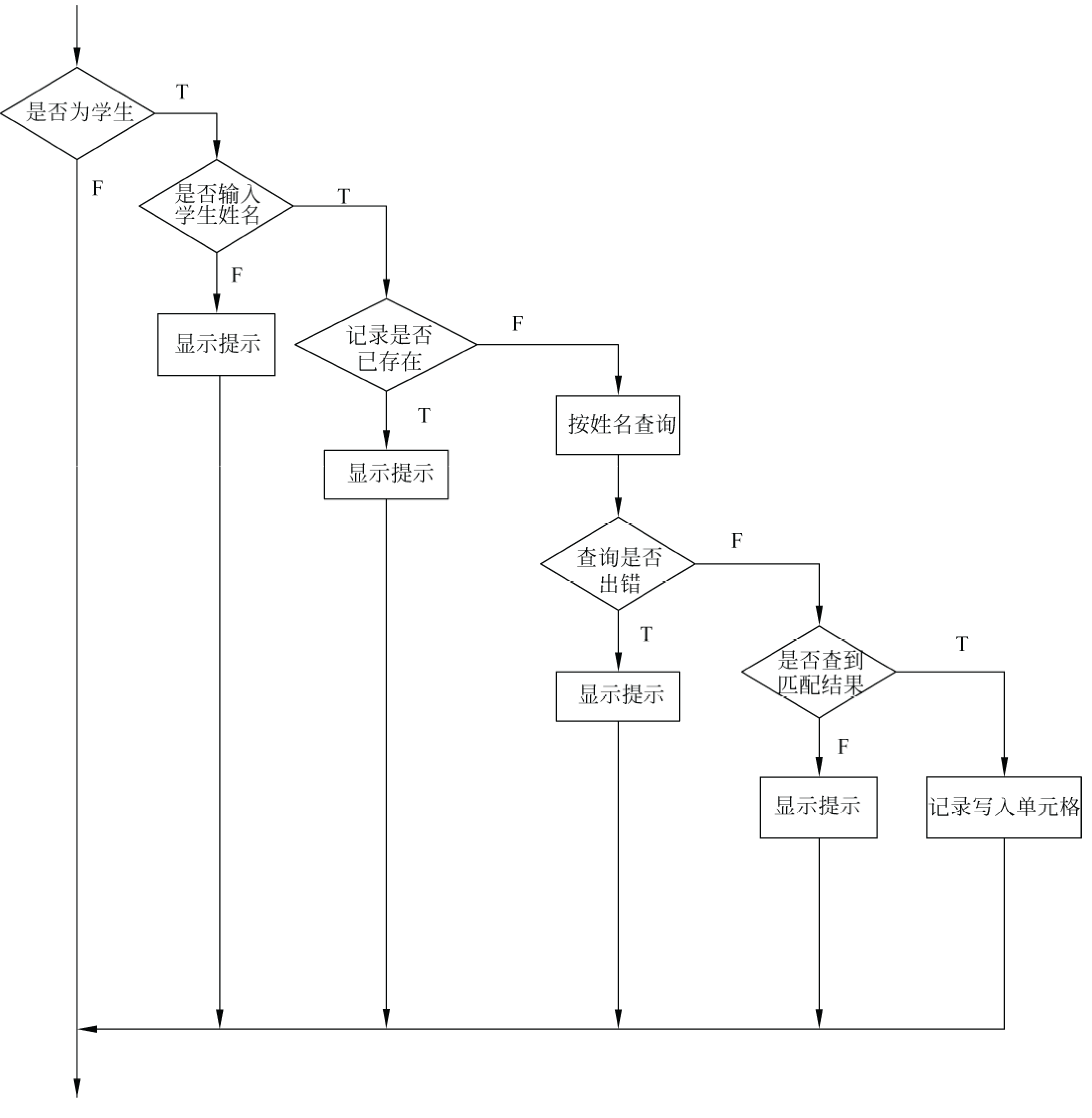


图 20.16　本段程序流程图

在“代码”窗口中为控件添加事件代码，代码如下所示：

```
Private Sub CommandButton1 Click()
    Dim n As Integer, i As Integer
    If ComboBox1.Value = "学生" Then                    '如果选择的是“学生”
        If Not (TextBox1.Text = Empty) Then             '如果输入了姓名
            If Sheet1.Range("a3") <> TextBox1.Text Then
                                                        '如果不是重复记录
                On Error GoTo NoMatch                   '跳到错误处理程序
                n = WorksheetFunction.Match(TextBox1.Text,
                Sheet2.Columns(1), 0)                   '查询记录
                If n <> 0 Then                          '查到了匹配的记录
                    Sheet2.Rows(n).Copy Destination:=
                    Sheet1.Range("a3")                  '记录复制到工作表中
                    i= Sheet4.Range("a1048576").End(xlUp)
                    .Row+1                              '获得第一个空白行行号
                    Sheet4.Cells(i, 1) = TextBox1.Text '记录学生姓名
                    Sheet4.Cells(i, 2) = Now            '记录学生登录时间
```

```
16                 Sheet1.Activate                 '激活“学生查询结果”表
17                 Exit Sub                        '退出过程
18             End If
19         Else
20             MsgBox "该学生记录已经存在！"         '提示记录已经存在
21             TextBox1.Text = ""                  '清空“姓名”文本框
22             TextBox1.SetFocus                   '“姓名”文本框获得焦点
23             Exit Sub                            '退出过程
24         End If
25     Else
26         MsgBox "没有输入学生姓名，请重新输入！" '提示没有输入学生姓名
27         TextBox1.SetFocus                       '“姓名”文本框获得焦点
28         Exit Sub                                '退出过程
29     End If
30 End If
31 NoMatch:
32     MsgBox "没有找到匹配的学生！"               '提示没有找到匹配学生
33     TextBox1.Text = ""                          '清空“姓名”文本框
34     TextBox1.SetFocus                           '“姓名”文本框获得焦点
35 End Sub
```

提示： 第 06 行代码使用 Match 函数来查询相匹配的内容，该函数返回一个 WorksheetFunction 对象变量，其值是匹配元素的位置。第 10 行代码使用 Copy 方法将输入姓名相匹配的姓名所在整行内容复制到工作表 Sheet1 中，Destination 参数指定了复制的目标地址。在使用 Match 函数对姓名进行查询时，如果没有相匹配的姓名，程序会出错，因此这里必须在其前面使用 On Error 语句捕获错误，出现错误后激活错误处理程序提示没有找到匹配内容。

（5）运行程序，单击“查询”按钮，如果是学生，则首先判断是否输入了姓名，以及在“学生成绩查询结果”表中是否存在同名的记录。如果没有输入姓名，程序给出提示，如图 20.17 所示。如果存在相同的记录，则提示记录已存在，以避免重复查询，如图 20.18 所示。



图 20.17　没有输入姓名时的提示信息



图 20.18　出现重复查询时的提示信息

当所有的检测都通过后，将按照输入的姓名进行查询，如图 20.19 所示。如果没有找到需要的学生，程序给出提示，如图 20.20 所示。

（6）当学生进行了查询后，需要将查询时使用的姓名和查询时间记录在工作表中，如图 20.21 所示。

这段代码比较简单，将“姓名”文本框的内容和当前的时间分别写入在 20.1 小节创建的“学生查询记录表”工作表的单元格中。这段代码放置于上一步的 CommandButton1（即“查询”按钮）的 Click 事件代码中，将其添加到 If Sheet1.Range("a3") <> TextBox1.Text Then

条件结构中，具体的程序代码如下所示：

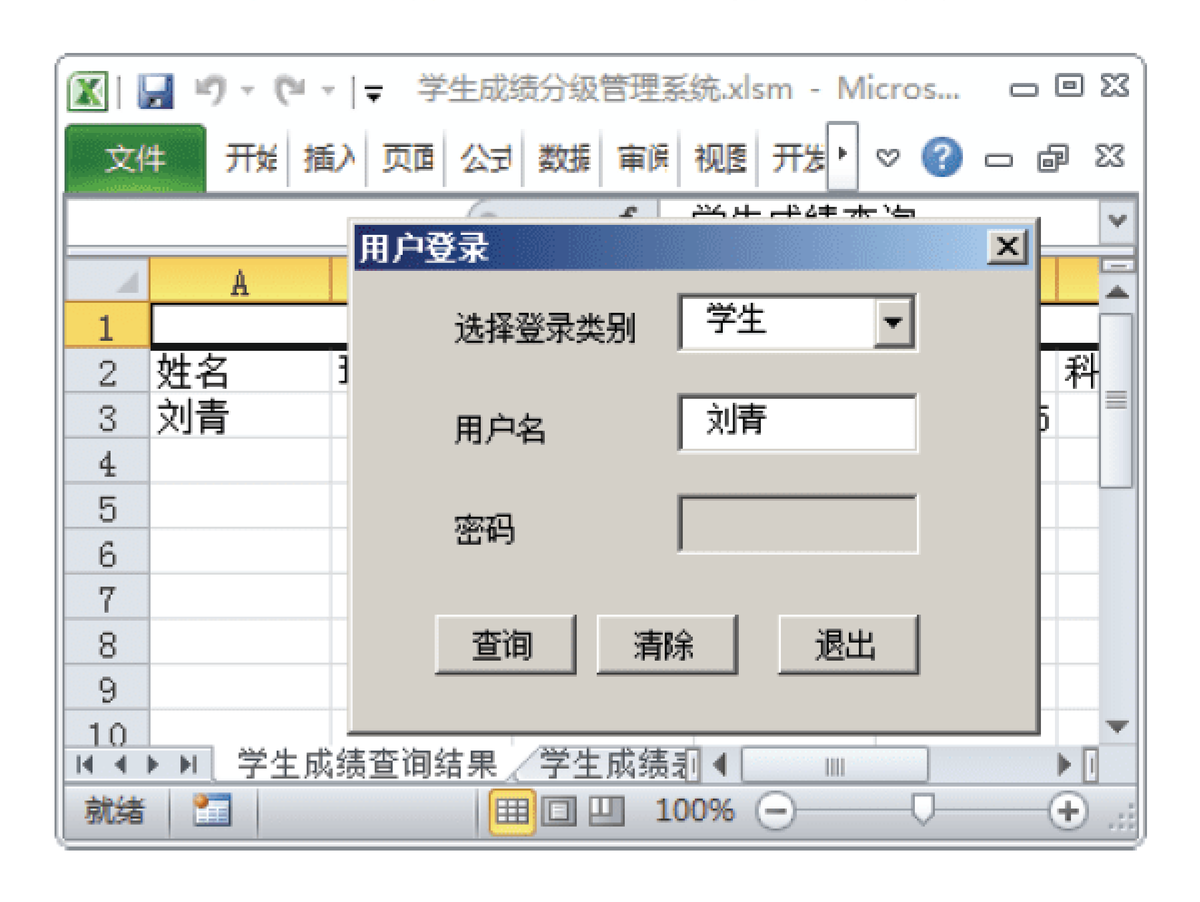


图 20.19　查询到的结果



图 20.20　没有查到学生时的提示

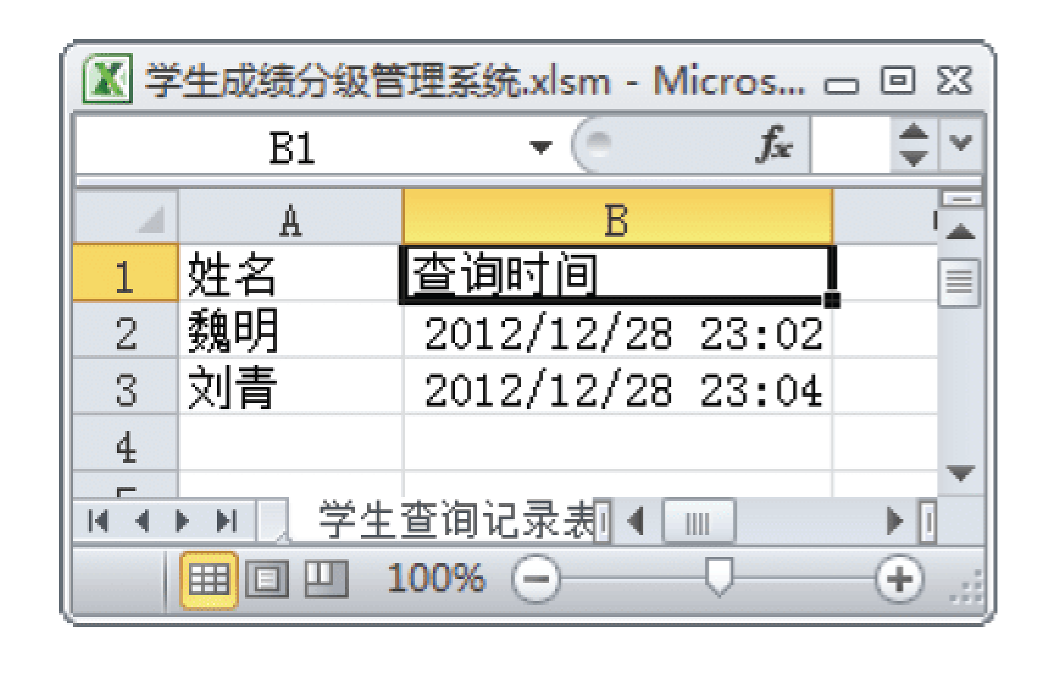


图 20.21　记录查询的姓名和查询时间

```
Cells(Sheets("学生查询记录表").Range("A1048576").End(xlUp)
.Row + 1, 1) = TextBox1.Text                                  '记录查询的姓名
Cells(Sheets("学生查询记录表").Range("A1048576").End(xlUp) _
.Row + 1, 2) = Now                                            '记录查询时间
```

提示： 使用 Sheets("学生查询记录表").Range("A1048576").End(xlUp) .Row + 1 语句指定紧接着上一个记录的空白行，这样能够保证新的记录紧接着老记录添加。Now 函数能获得当前系统的日期和时间。这段代码放在 If Sheet1.Range("a3") <> TextBox1.Text Then 条件结构中，能够保证只要输入了姓名的查询都将记录，不管这个姓名是否在成绩表中找到。

20.4　教师查询分数

为了对考试情况进行分析，教师在查询分数时往往需要同时使用多个条件，如按班级查询、查询语文和数学的分数都在 100 分以上的学生等。要实现这样的复杂条件下的数据查询，结合 Excel 的高级查找功能来编写程序是一个好的方法。

20.4.1　创建查询表

在工作簿中创建一个名为“学生成绩查询表”的工作表，该工作表作为高级查询的操作界面。工作表中将创建查询条件区域和查询结果区，同时为操作添加按钮控件。

（1）在 Excel 2010 窗口下方的任意一个工作表标签上右击，在弹出的快捷菜单中选择“插入”命令打开“插入”对话框。在对话框中选择“工作表”选项后单击“确定”按钮关闭“插入”对话框，如图 20.22 所示。此时，工作表中会插入一个新的工作表，将工作表命名为“学生成绩查询表”。

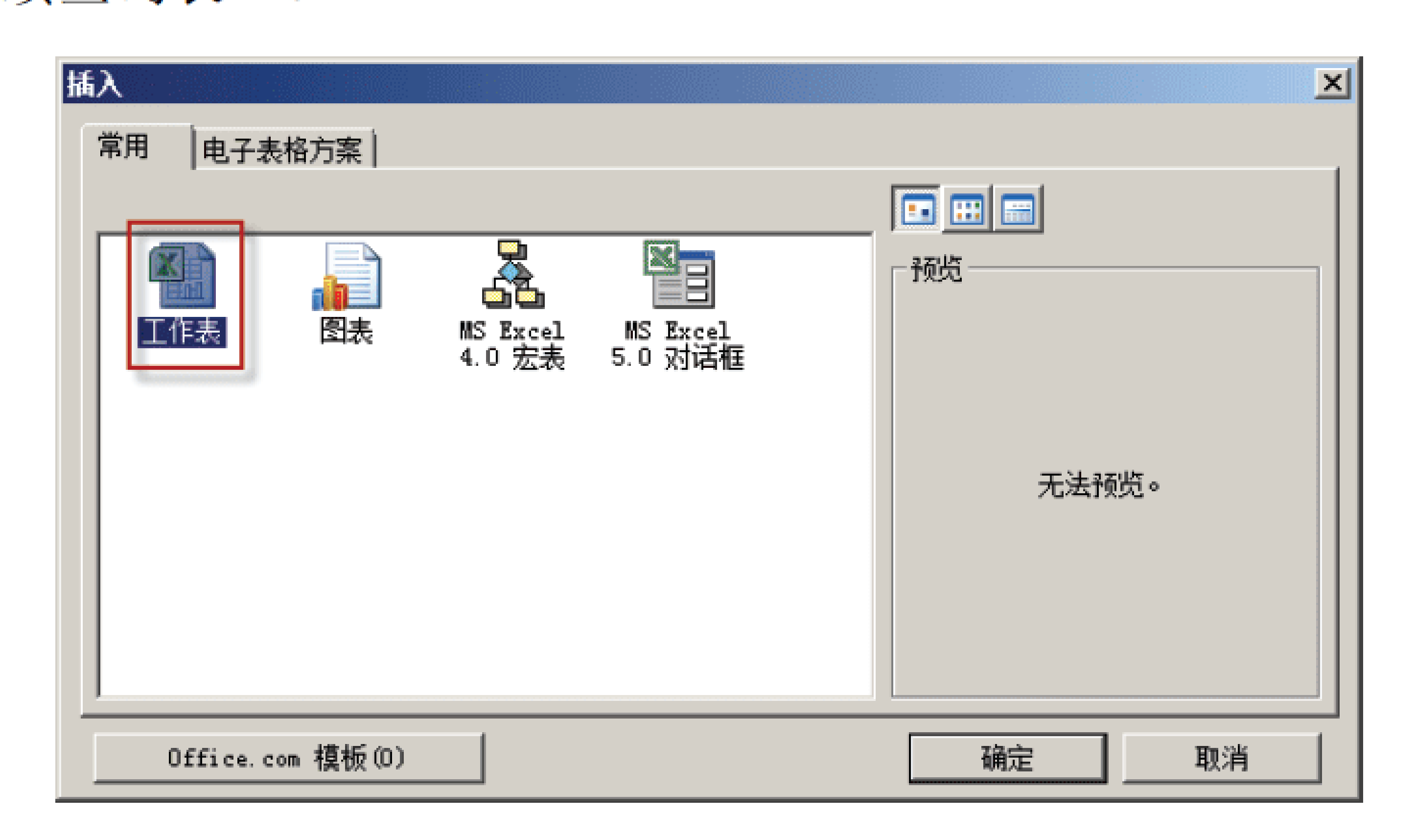
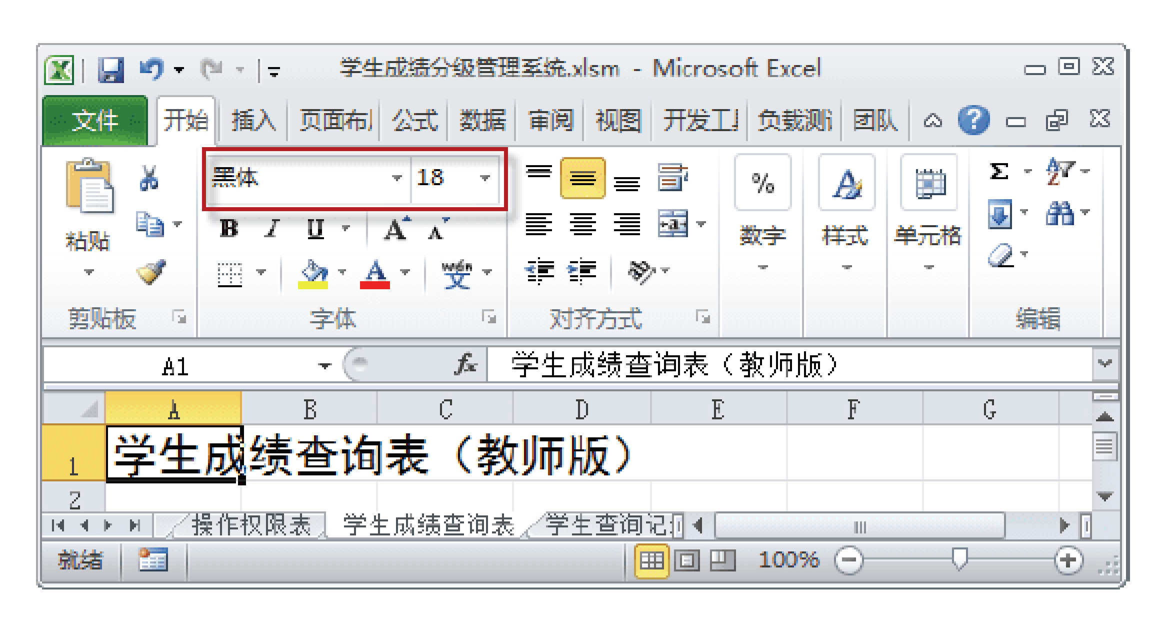


图 20.22　选择插入工作表

（2）在工作表的 A1 单元格中输入工作表标题，设置标题文字的字体和大小，如图 20.23 所示。拖动光标框选 A1 至 G1 单元格，标题文字设置为“合并后居中”，如图 20.24 所示。



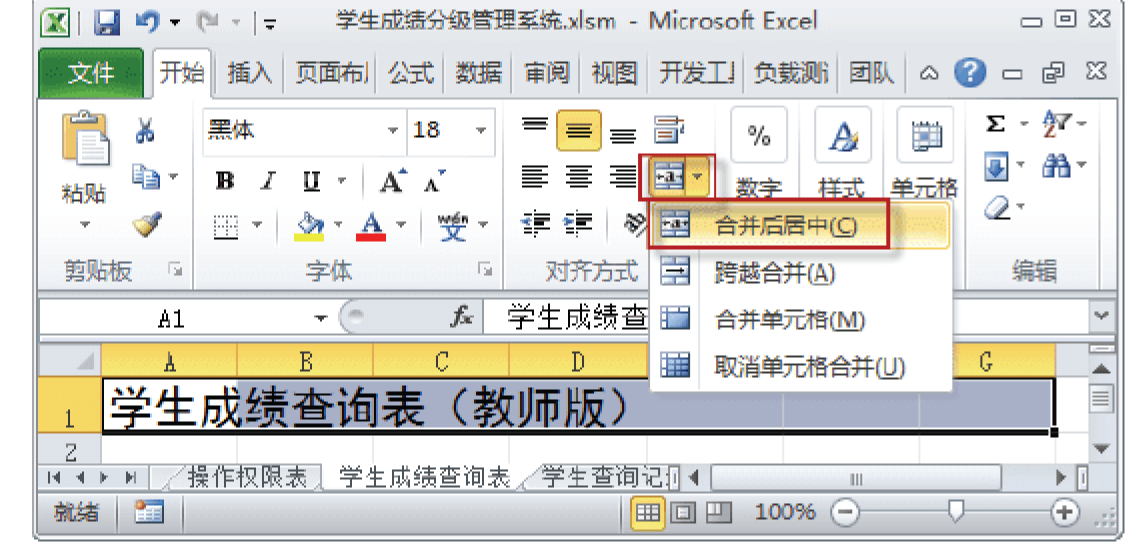




图 20.23　设置标题字体和字号　　　　图 20.24　将标题文字在合并单元格中居中

（3）拖动光标同时选择 A2 至 G2 单元格，将单元格合并，如图 20.25 所示。单击“填充颜色”按钮上的下三角按钮，在打开的菜单中选择需要绿色填充合并的单元格区域，如图 20.26 所示。将文字颜色设置为白色，在单元格区域中输入文字，如图 20.27 所示。

提示： 选择颜色填充后，Excel 会自动记录这次使用的颜色，当下次需要使用相同的颜色时，只需要单击“填充颜色”按钮即可。

（4）选择 A3 至 G4 单元格区域后右击，在弹出的快捷菜单中选择“设置单元格格式”命令，打开“设置单元格格式”对话框。在“边框”选项卡中设置单元格边框样式，如图

20.28 所示。

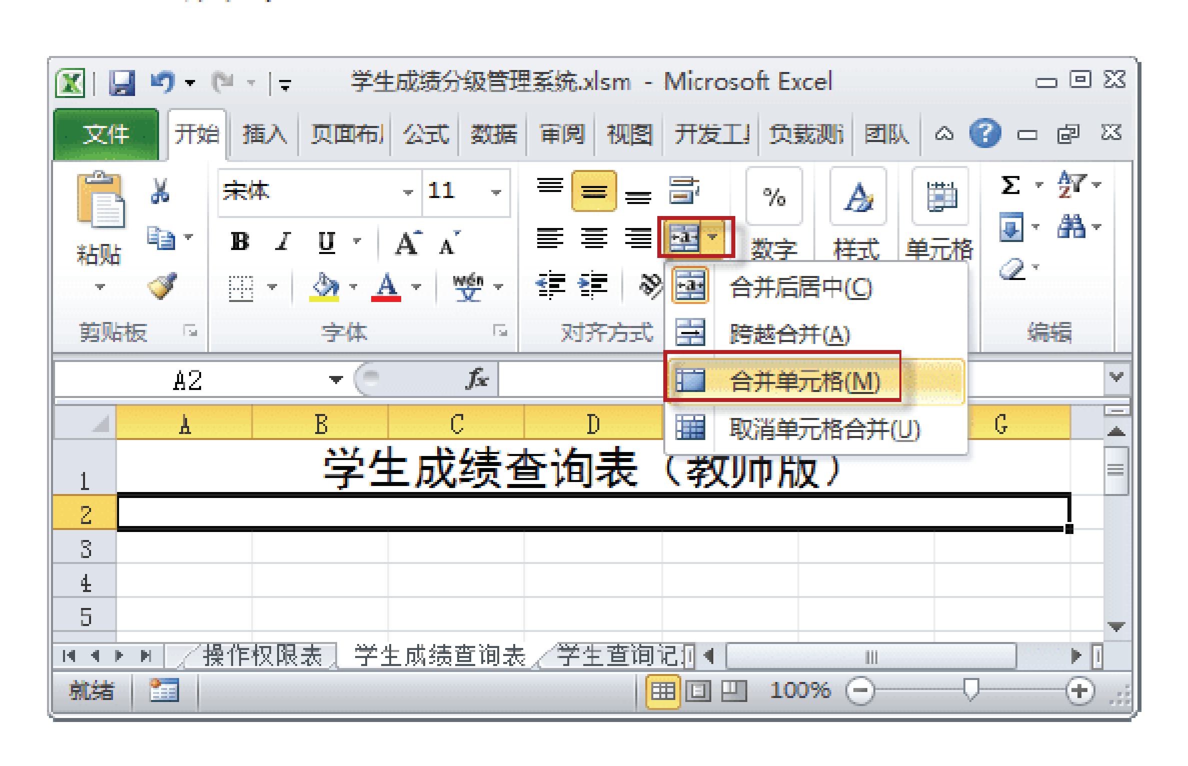


图 20.25　合并单元格

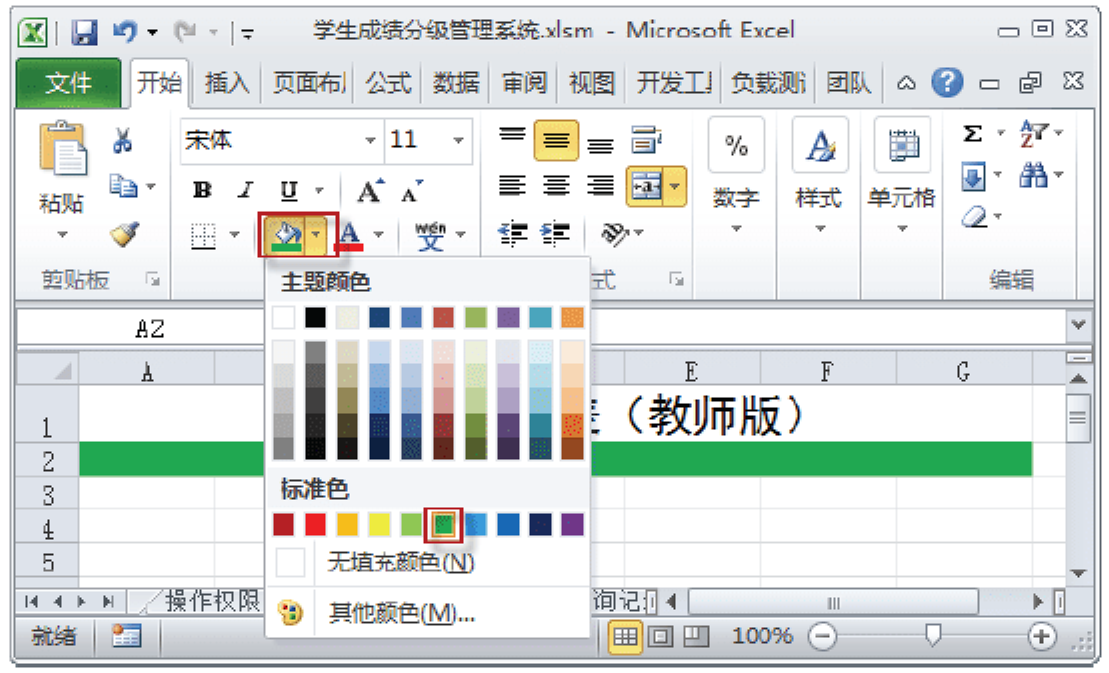


图 20.26　填充合并的单元格区域

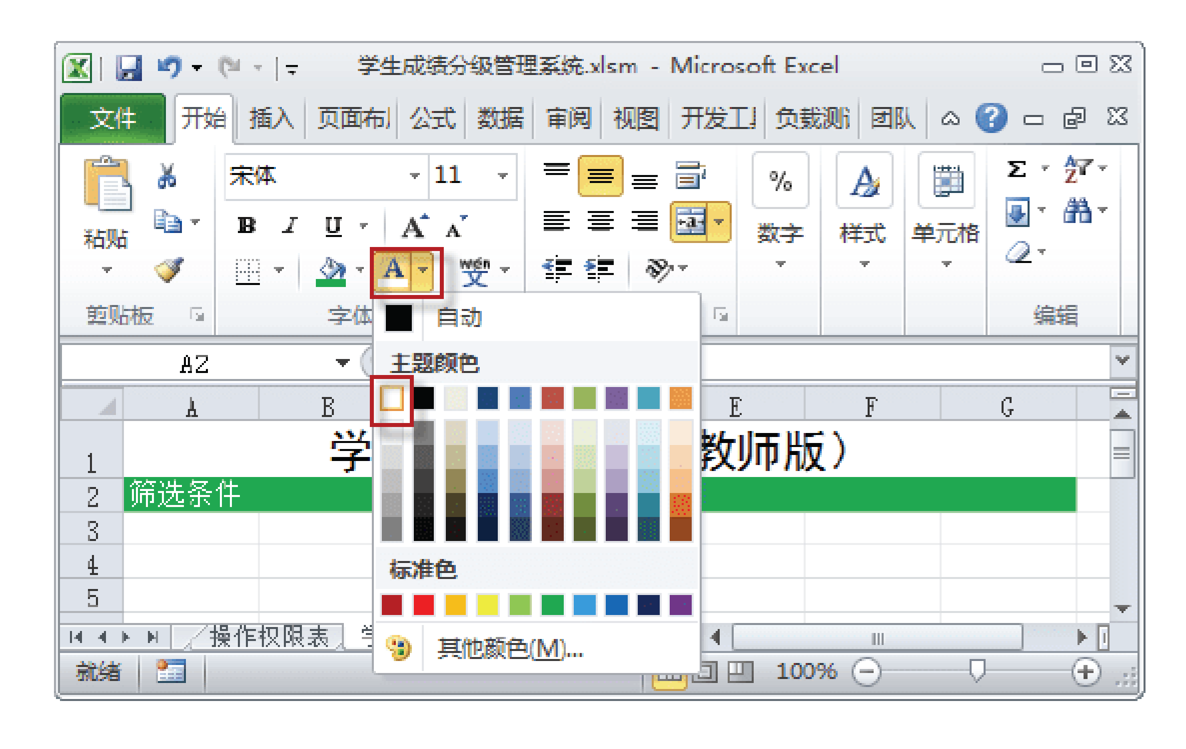


图 20.27　输入白色文字

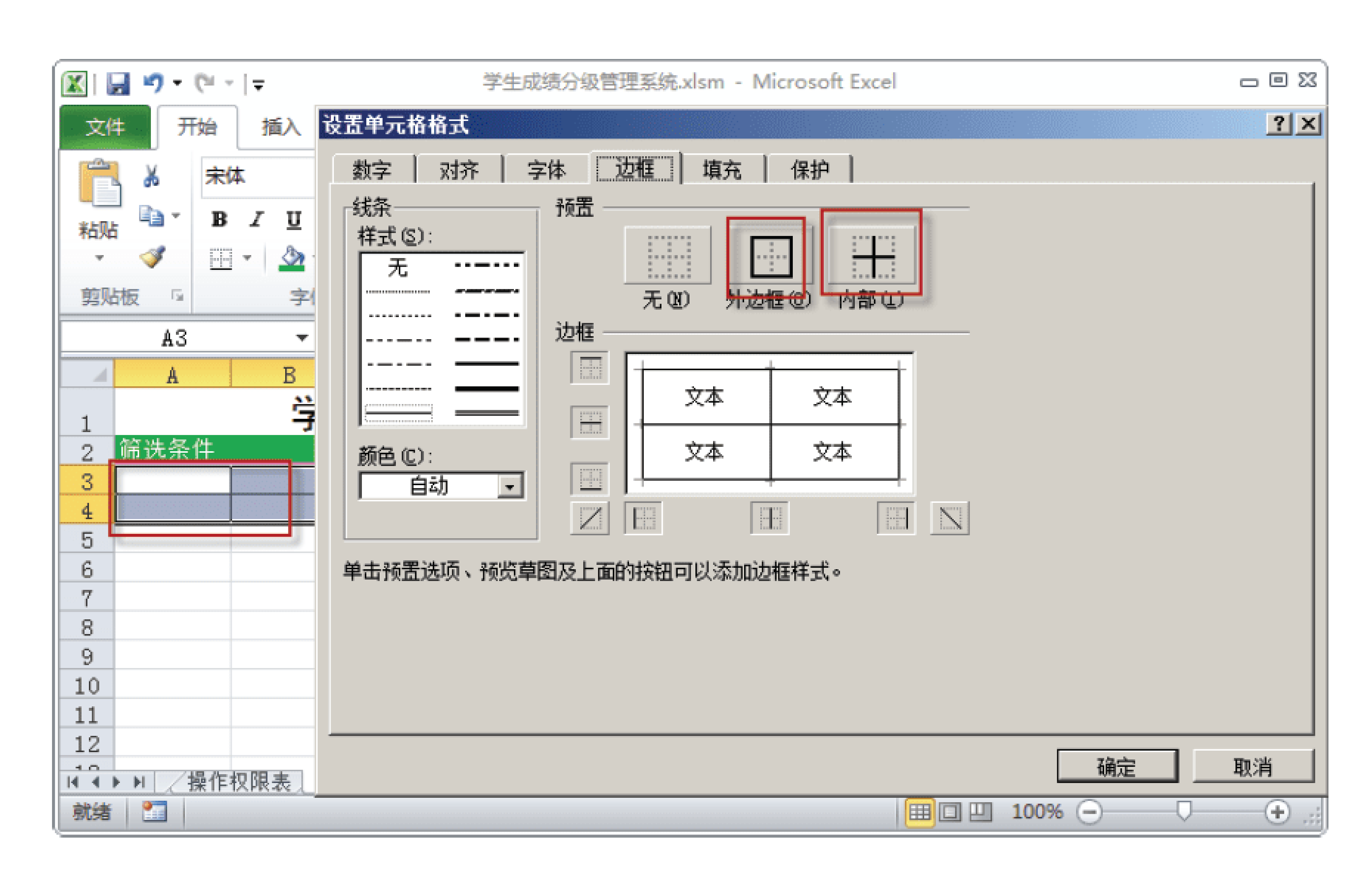


图 20.28　设置单元格边框

（5）单击功能区“填充颜色”按钮上的下三角按钮，在打开的菜单中选择“其他颜色”命令，如图 20.29 所示。此时将打开“颜色”对话框，在对话框的“自定义”选项卡中，输入 R、G 和 B 颜色值设置颜色。单击“确定”按钮关闭“颜色”对话框，选择单元格填充颜色，如图 20.30 所示。

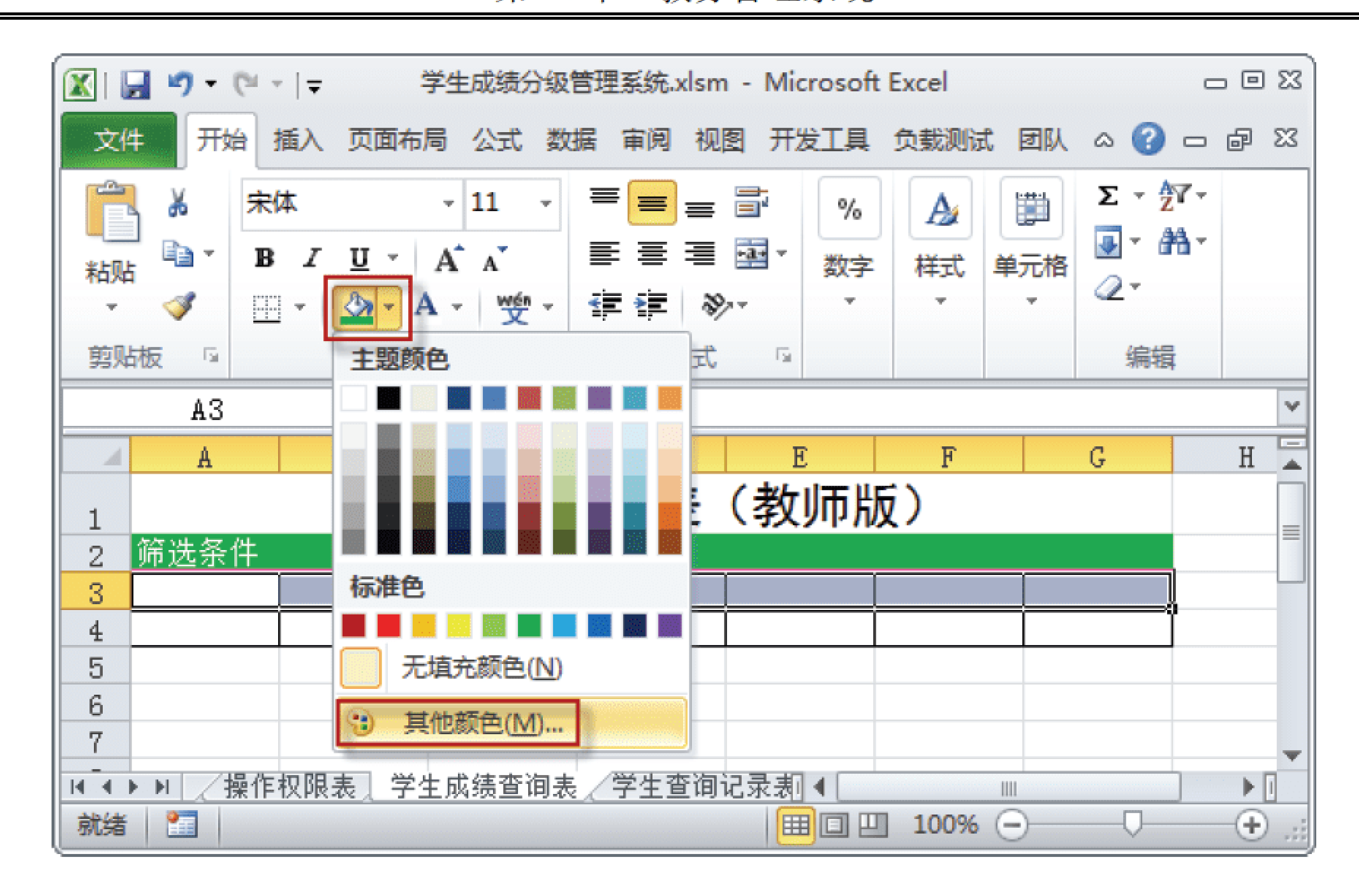


图 20.29　选择“其他颜色”命令

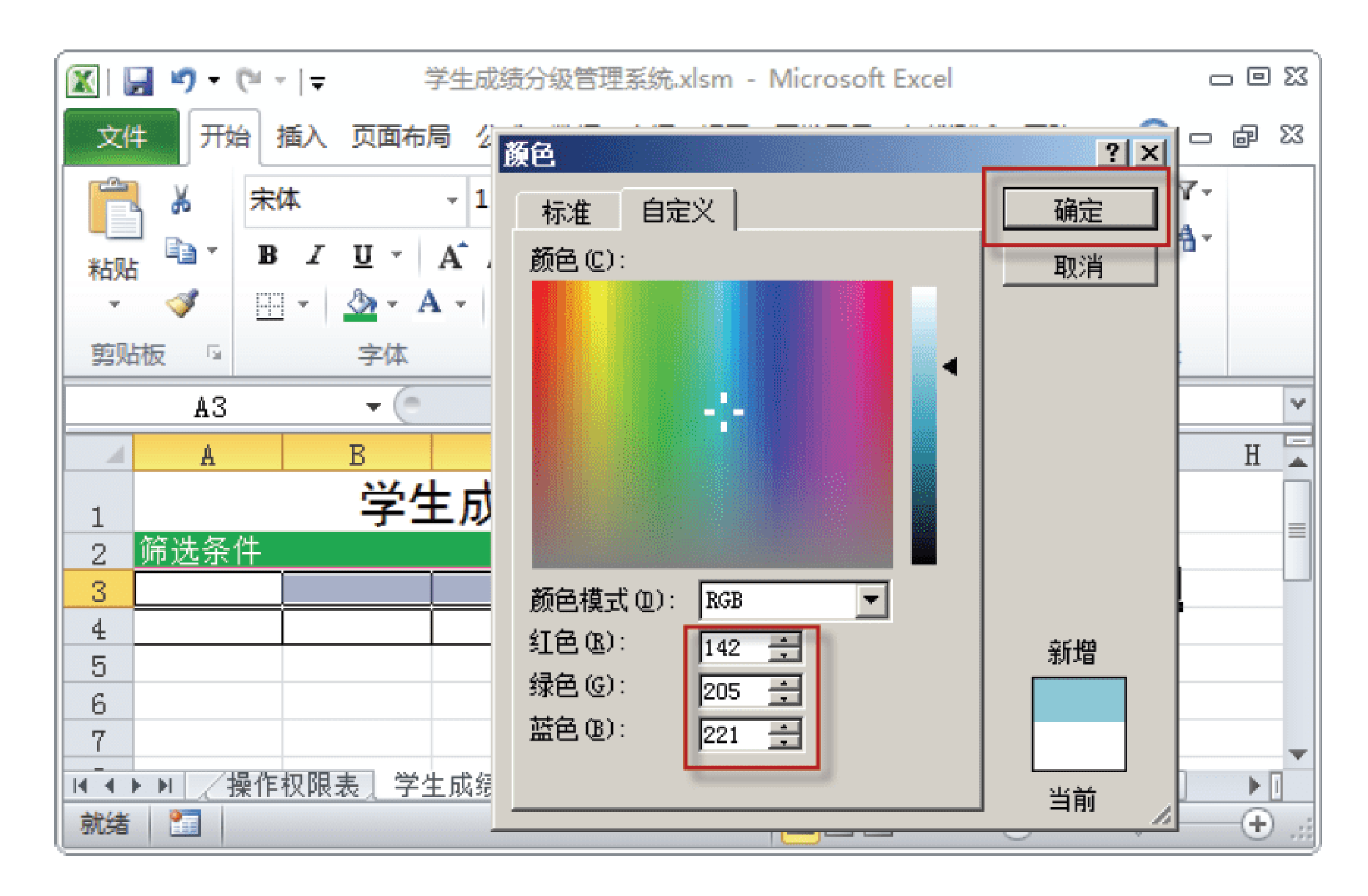


图 20.30　使用自定义颜色填充单元格区域

提示： 在“自定义”选项卡中，“红色”、“绿色”和“蓝色”增量框用于设置 RGB 颜色值，其输入值均在 0～255 之间。同时，在“颜色”框中单击可以直接选择颜色。

（6）在 A3 至 G3 单元格中分别输入标题文字，如图 20.31 所示。这个表格用于输入筛选的条件。在“学生成绩查询表”工作表中，从 A6 单元格开始向下的单元格区域将用于显示成绩筛选结果。同时在 I5 至 I6 单元格将用于显示查询到的记录总数。使用和上面相同的方法制作提示标题，并为单元格添加颜色，如图 20.32 所示。

（7）选择功能区的“开发工具”选项卡，单击“插入”按钮，在获得的菜单中选择“表单控件”中的“按钮”控件，拖动光标在工作表中绘制一个按钮，如图 20.33 所示。在按

钮上单击后修改按钮标题，如图 20.34 所示。该按钮用于启动数据的筛选。

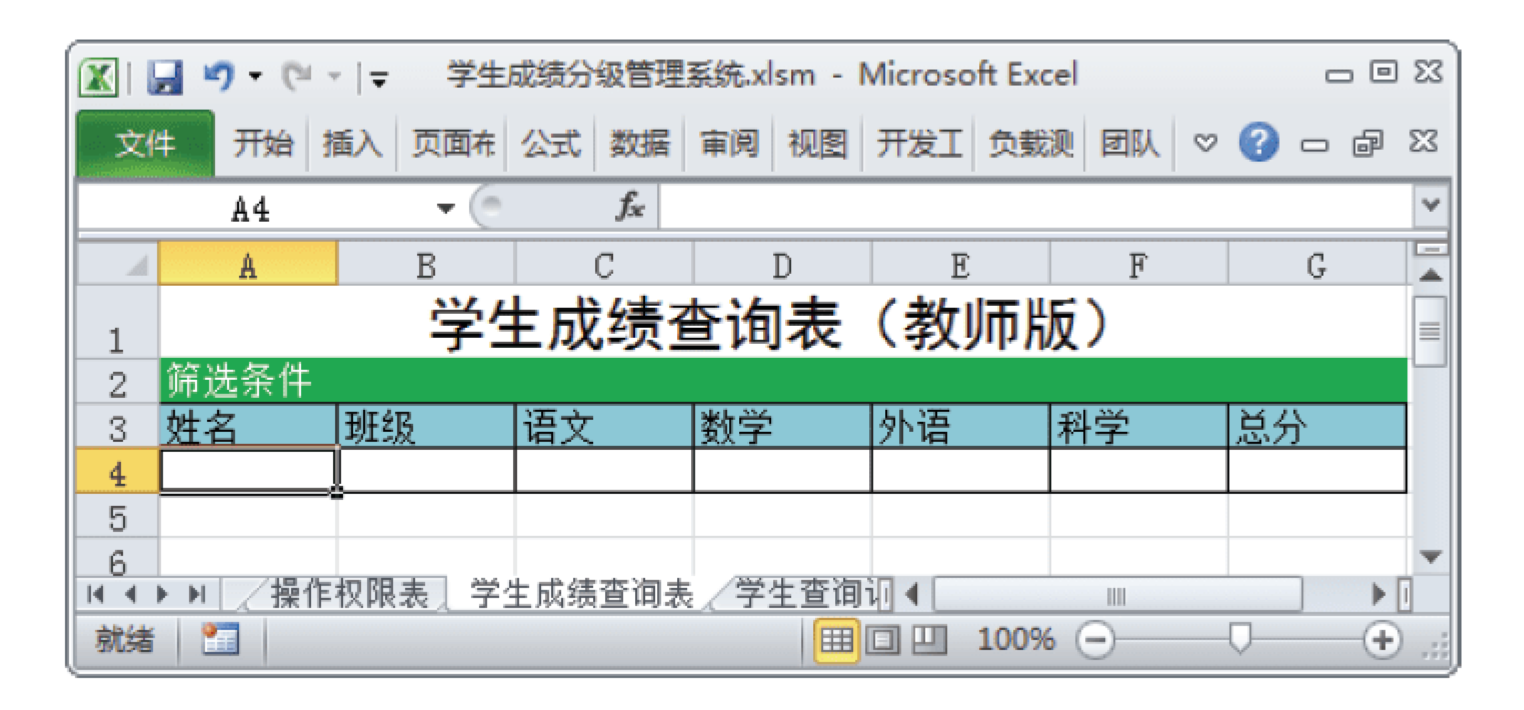


图 20.31　向单元格写入文字

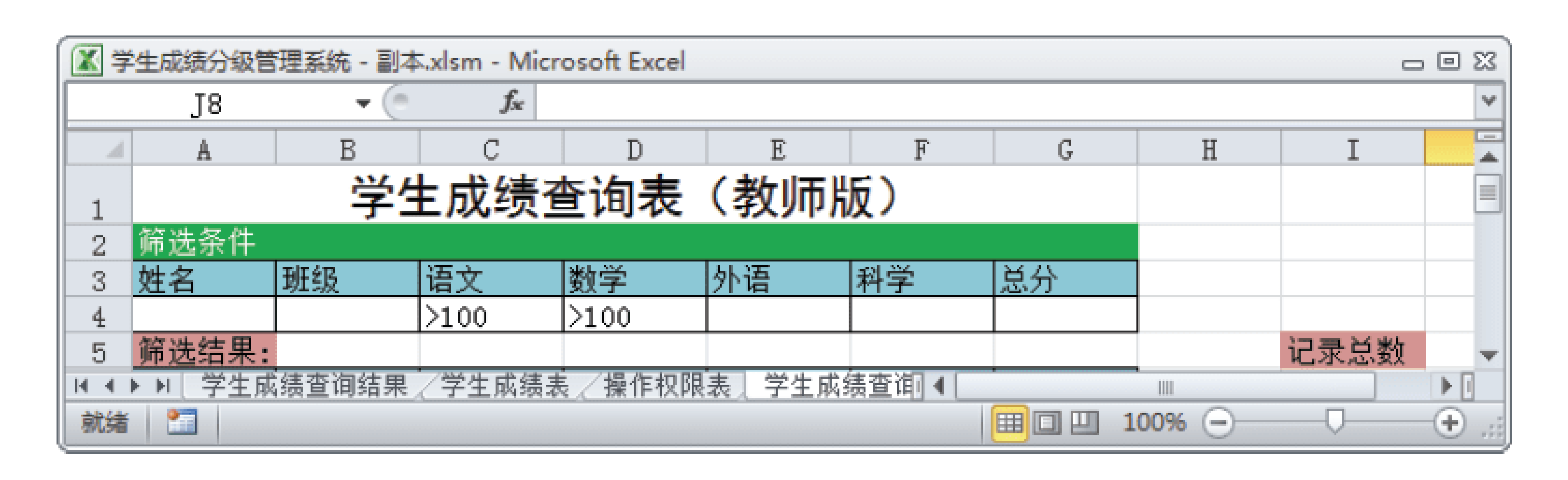


图 20.32　创建其他提示文字

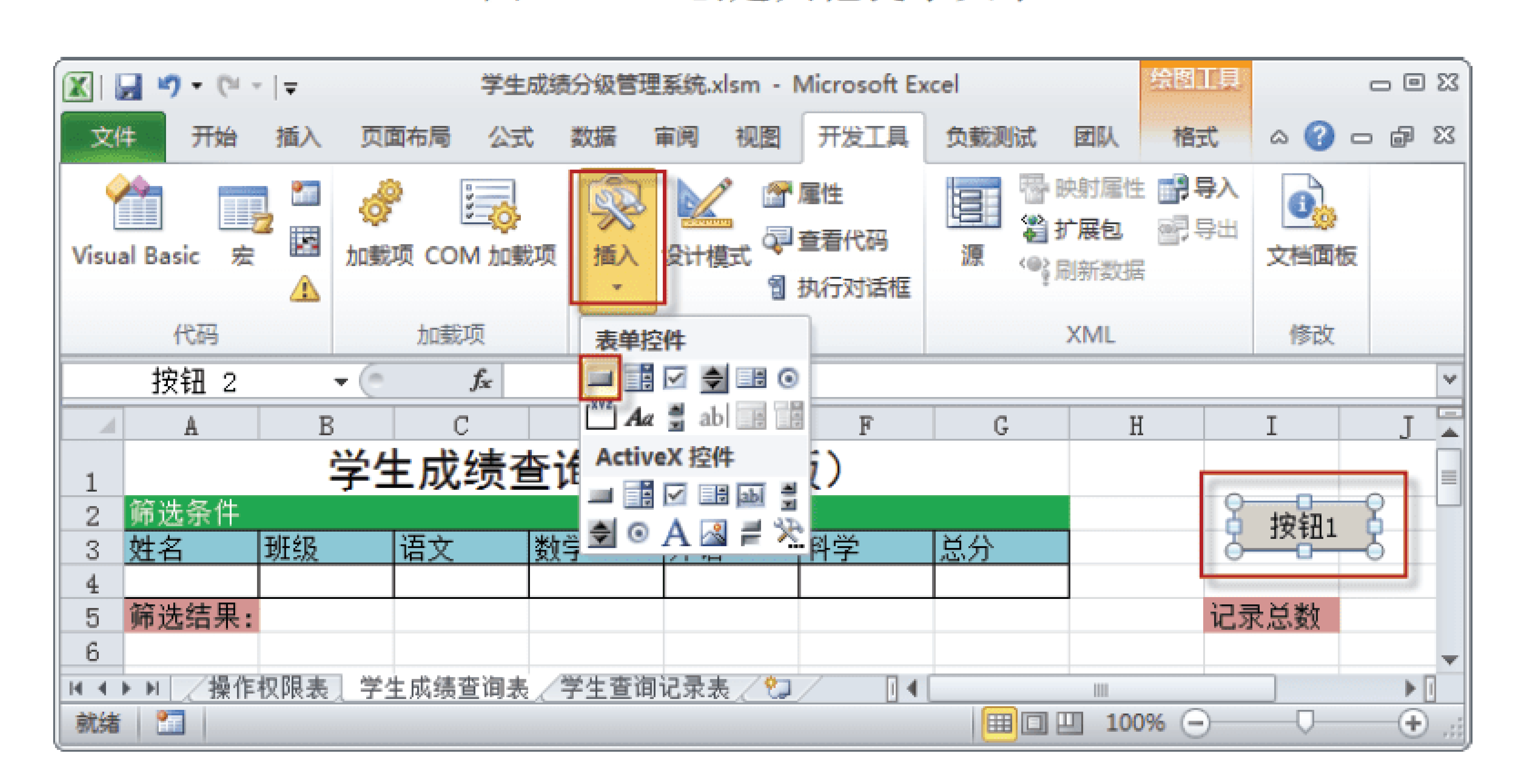


图 20.33　绘制一个“命令按钮”控件

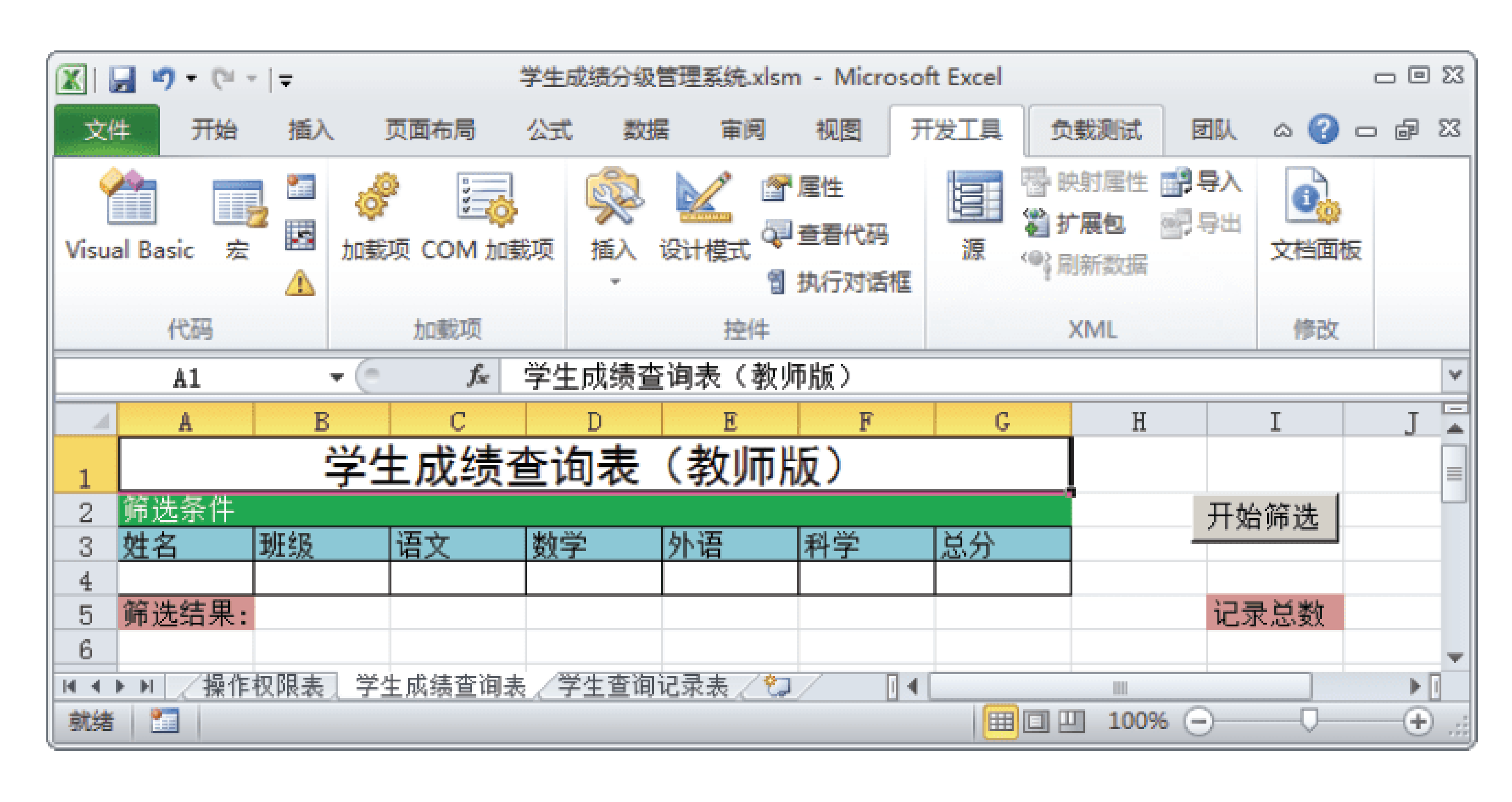


图 20.34　修改按钮标题

（8）使用相同的方法再添加一个“按钮”控件，将其标题修改为“回到登录窗口”。单击该按钮能够重新打开登录窗口。至此，查询表设计制作完成。完成后的查询表如图 20.35 所示。

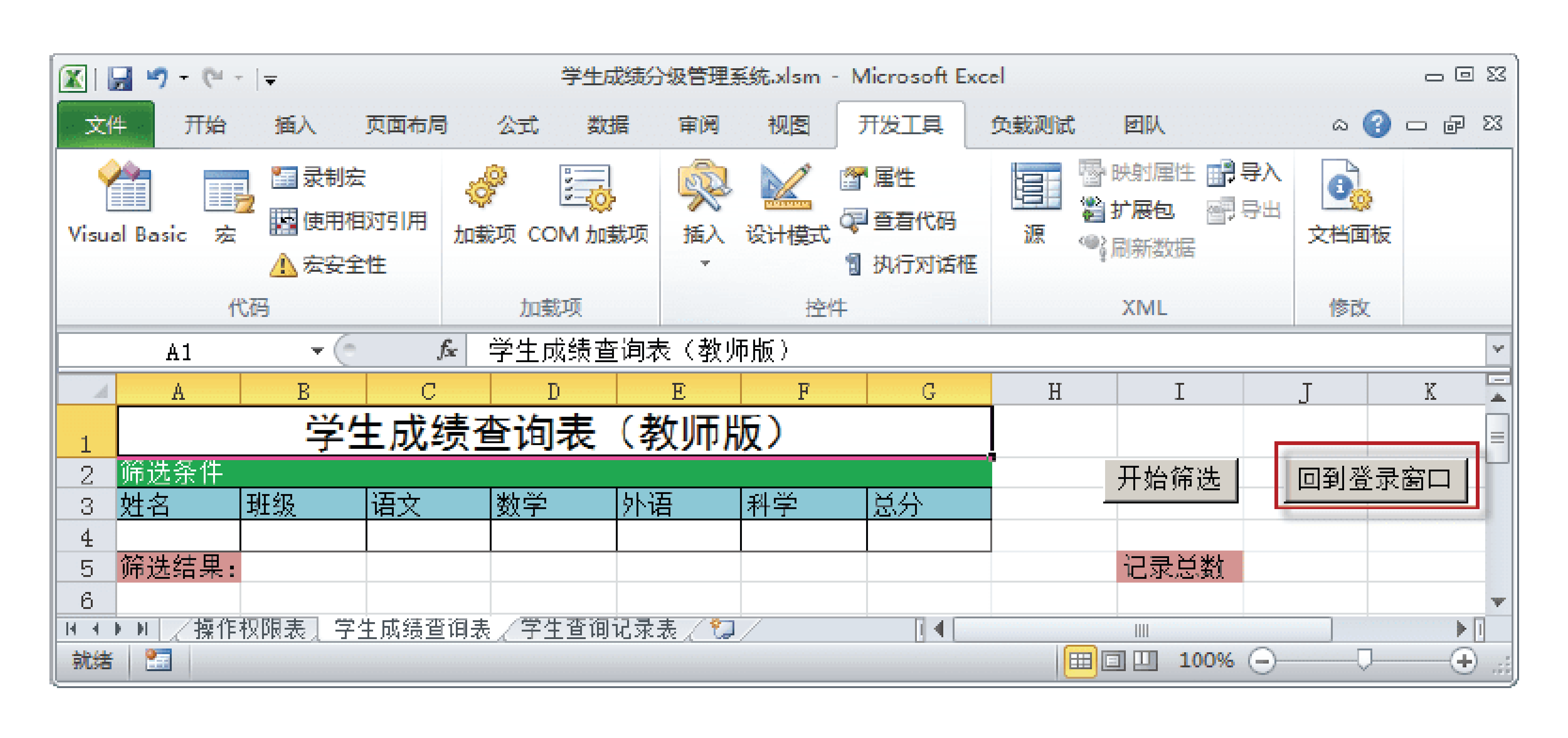


图 20.35　制作完成后的工作表

注意：在添加“按钮”控件时，Excel 会自动弹出“指定宏”对话框要求为按钮指定一个宏。此时只需要单击对话框中的“取消”按钮关闭对话框即可。指定按钮启动的宏代码将在下面编写。

20.4.2　实现分数查询

在 Excel VBA 中，使用高级筛选功能来查找同时满足多个条件的记录十分简单，直接使用 AdvancedFilter 方法即可。在查询完成后需要回到登录窗口，可通过编程使登录窗体重新显示来实现这一功能。

（1）切换到 VBA 编辑器，在工程资源管理器中添加一个模块，打开模块的“代码”窗口，创建一个名为“分数查询”的过程。该过程代码实现 3 个功能，首先根据设置的条件来查询记录并将记录写入指定单元格。接着将记录的条数写入指定的单元格，最后设置写有记录单元格的样式。“分数查询”过程代码如下所示：

```
Sub 分数查询()
    Dim n As Integer, m As Integer
    Application.ScreenUpdating = False                    '关闭屏幕刷新
    Sheets("学生成绩表").Range("A1:G500").AdvancedFilter _
    Action:=xlFilterCopy,CriteriaRange:=Sheets("学生成绩查询表") _
    .Range("A3:G4"), CopyToRange:=Sheets("学生成绩查询表").Range
    ("A6:G6") _
    , Unique:=False                                       '根据条件进行查询并复制结果
    m = Sheets("学生成绩查询表")
    .Range("a1048576").End(xlUp).Row                      '获得工作表中非空行数
    n = m - 6                                             '去掉标题区和条件区域
    Range("I6") = n                                       '将记录条数写入指定单元格
    With Range("A6:G" & m)                                '设置写有记录单元格的样式
        .Interior.Color = RGB(142, 205, 220)              '设置单元格填充颜色
        .Font.Bold = True                                 '文字加粗
        .Font.ColorIndex = 2                              '设置文字颜色
        .Borders.ColorIndex = vbBlack                     '设置边框颜色
```

```
17          .Borders.LineStyle = xlContinuous          '设置边框线条样式
18          .Borders.Weight = xlThin                   '设置边框宽度
19      End With
20      Application.ScreenUpdating = True              '开启屏幕刷新
21  End Sub
```

提示： 在这段代码中，第 04～07 行代码实现对成绩表的高级筛选，其中 Action:= xlFilterCopy 表示复制筛选结果，CopyToRange:=Sheets("学生绩查询表") .Range ("A6:G6")表示筛选结果复制的区域，CriteriaRange:=Sheets("学生成绩查询表") .Range("A3:G4")表示筛选条件所在区域。在工作表中，A1 至 A5 单元格区域不是筛选结果区域，A6 单元格显示筛选结果的标题，因此真正学生分数记录的条数应该是所有第一列的非空单元格数减去 6，这也就是第 10 行代码变量 m 需要减去 6 的原因。

（2）在模块 1 中创建一个名为“返回登录窗体”的过程，该过程代码执行时将重新激活“学生成绩查询结果”工作表，并使刚才使用的“学生成绩查询表”不可用，同时使登录窗体再次显示。“返回登录窗体”的过程代码如下所示：

```
Sub 返回登录窗体()
    Sheets("学生成绩查询结果").Activate          '激活“学生成绩查询结果”工作表
    Sheets("学生成绩查询表").Visible=False       '“学生成绩查询表”不可见
    UserForm1.Show                               '显示登录窗体
End Sub
```

（3）切换到“学生成绩查询表”工作表，在“开始筛选”按钮上右击，在弹出的快捷菜单选择“指定宏”命令。在打开的“指定宏”对话框中选择“分数查询”选择，如图 20.36 所示。单击“确定”按钮，关闭该对话框为控件添加宏。采用相同的方法将“返回登录窗体”过程指定给“回到登录窗口”控件。

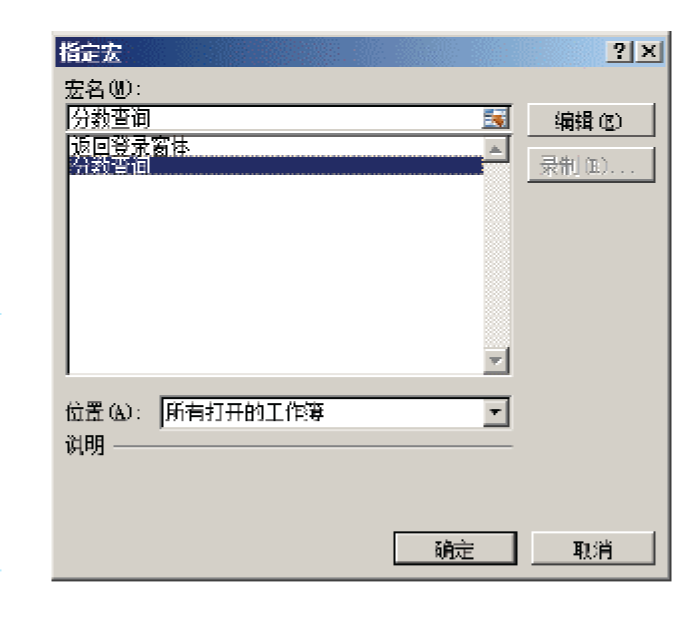


图 20.36　为控件指定宏

（4）至此，教师成绩查询表制作完成。首先在条件区域中输入筛选条件，然后单击“开始筛选”按钮，在工作表中将显示筛选获得的记录，如图 20.37 所示。单击“回到登录窗口”按钮，将显示“学生成绩查询表”和登录窗口，如图 20.38 所示。

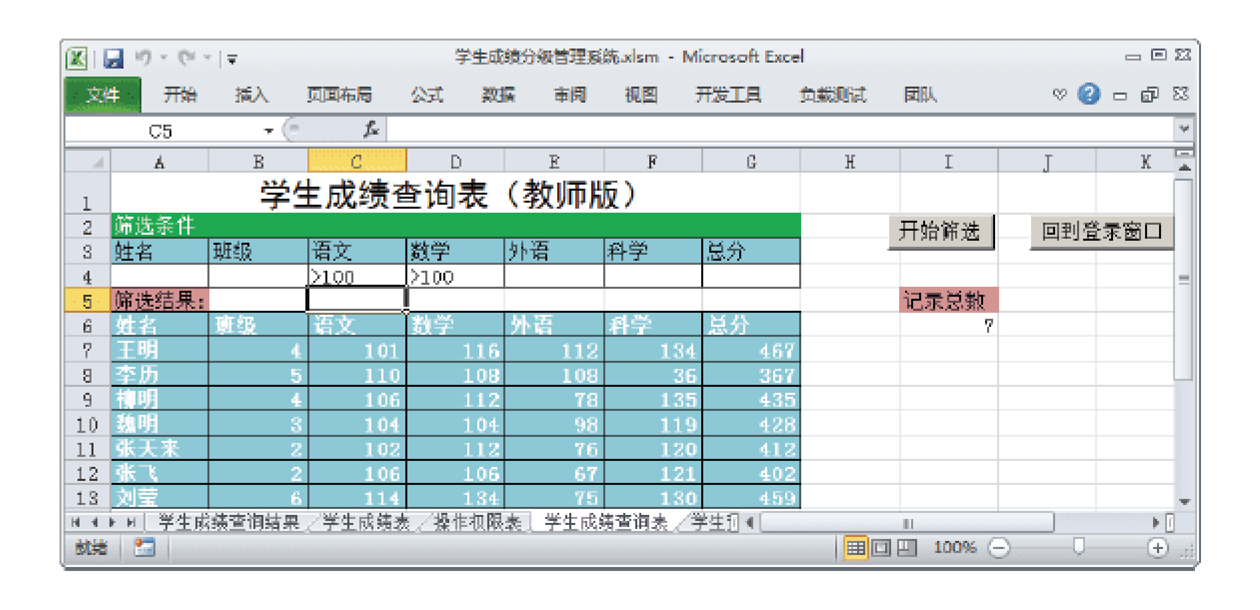


图 20.37　工作表中显示查询结果

（5）最后实现由登录窗口进入“学生成绩查询表”。这里，功能的实现是通过“登录”按钮的事件代码的编写来实现，将下面的代码添加到“登录”按钮的 Click 事件代码的最后即可，程序的流程示意图如图 20.39 所示。

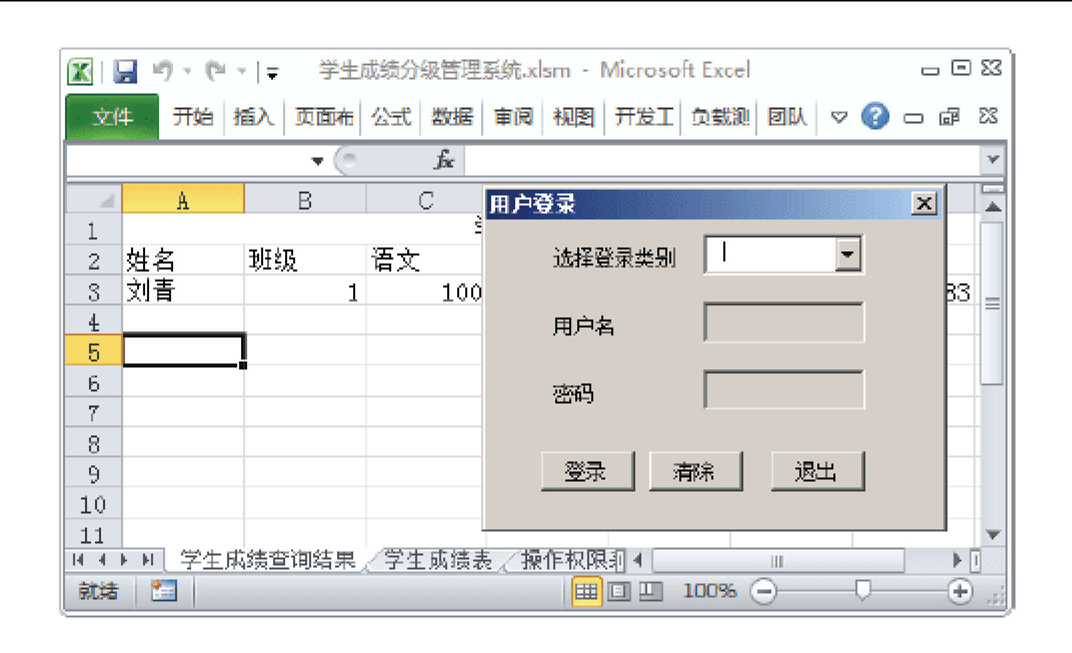


图 20.38　重新显示表格与登录窗口

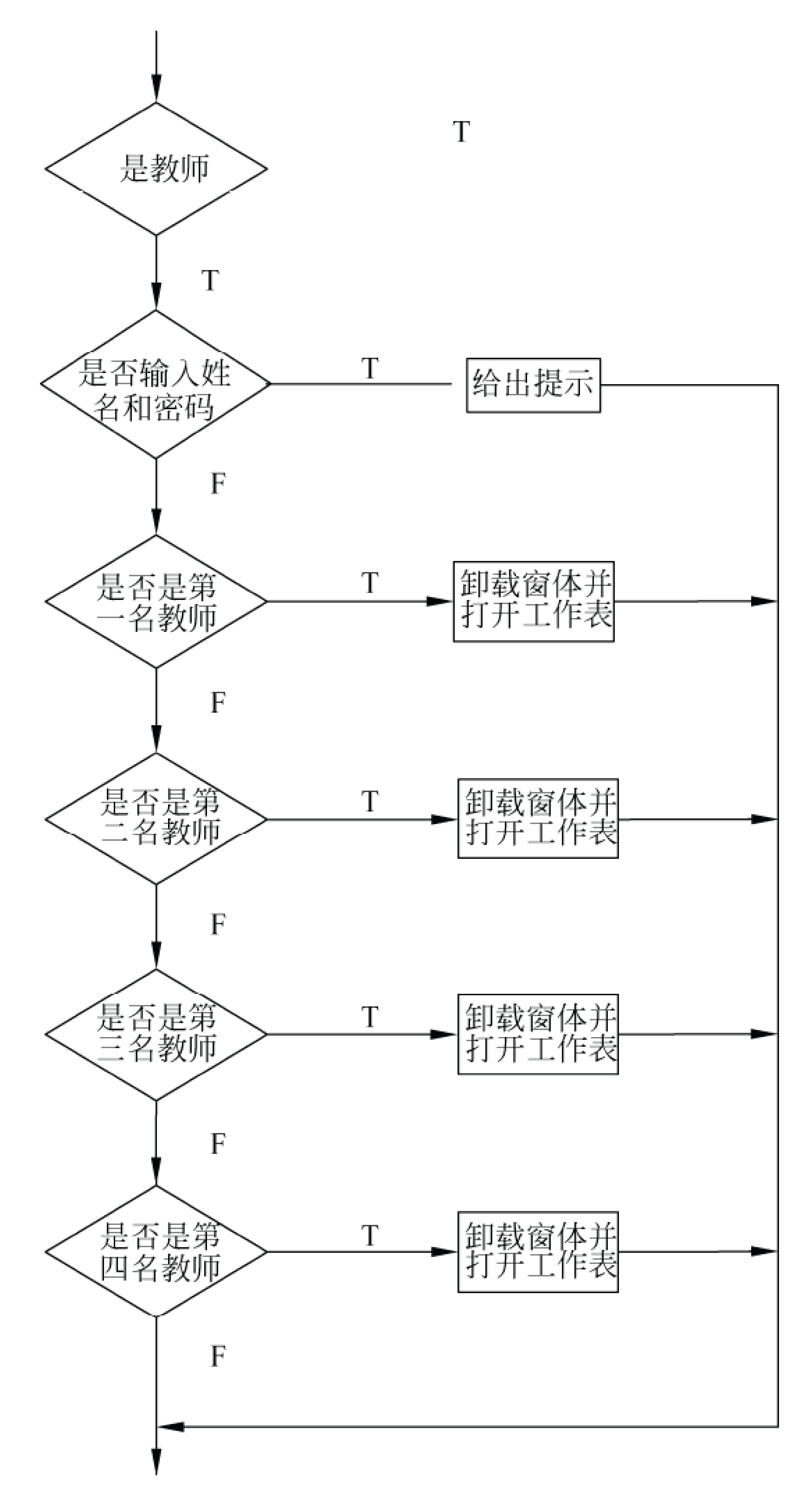


图 20.39　程序流程

详细的程序代码如下所示：

```
    If ComboBox1.Value = "教师" Then                    '如果选择的是教师
        If Not (TextBox1.Text = Empty) Or Not
        (TextBox2.Text = Empty) Then                    '如果输入了姓名和密码
            If TextBox1.Text = Sheets("操作权限表") _
```

```
                .Range("B2") And TextBox2.Text = _
                 Sheets("操作权限表").Range("C2") Then    '判断是否是第一个教师
                    Unload Me                            '如果是则卸载窗体
                    Sheets("学生成绩查询表").Activate '激活"学生成绩查询表"
                    Exit Sub                             '退出过程
                ElseIf TextBox1.Text = Sheets("操作权限表").
                Range("B3") And TextBox2.Text = _
                Sheets("操作权限表").Range("C3") Then    '判断是否是第二个教师
                    Unload Me                        '如果是则卸载窗体
                    Sheets("学生成绩查询表").Activate'激活"学生成绩查询表"
                    Exit Sub                         '退出过程
                ElseIf TextBox1.Text = _
                Sheets("操作权限表").Range("B4") And TextBox2.Text _
                = Sheets("操作权限表").Range("C4") Then '判断是否是第三个教师
                    Unload Me                        '如果是则卸载窗体
                    Sheets("学生成绩查询表").Activate'激活"学生成绩查询表"
                    Exit Sub                         '退出过程
                ElseIf TextBox1.Text =
                Sheets("操作权限表").Range("B5") And TextBox2.Text _
                = Sheets("操作权限表").Range("C5") Then '判断是否是第四个教师
                    Unload Me                            '如果是则卸载窗体
                    Sheets("学生成绩查询表").Activate      '激活"学生成绩查询表"
                    Exit Sub                             '退出过程
                Else
                    MsgBox "非指定的教师，您无权查看工作表！"'如果不是已有教师提示
                    Exit Sub                         '退出过程
                End If
            Else
                MsgBox "请输入姓名和密码"                '提示输入姓名和密码
                TextBox1.SetFocus                    '"姓名"文本框获得焦点
                Exit Sub
            End If
        End If
```

提示： 程序运行时，单击“登录”按钮，如果选择的是“教师”选项，则首先判断是否输入姓名和密码，如果没有输入，则给出提示。如果已经输入，则判断输入的姓名和密码是否与“操作权限”工作表中第一个教师的姓名和密码同时匹配，如匹配，则卸载窗体，并激活“学生成绩查询表”。对“操作权限”工作表中的每个教师姓名和密码都比较一次；如果没有匹配的，判断为非指定教师，将无权打开工作表。

（6）运行程序，当在“操作权限”表中指定的教师在登录窗口中输入了姓名和密码后，单击“登录”按钮即可进入“学生成绩查询表”，查询学生成绩。当然，程序同样会对用户的合法性进行判断，如果密码或姓名不是“操作权限”工作表中的指定内容，则会给出提示，并退出过程，如图 20.40 所示。如果没有输入姓名或密码，则程序提示用户输入，如图 20.41 所示。

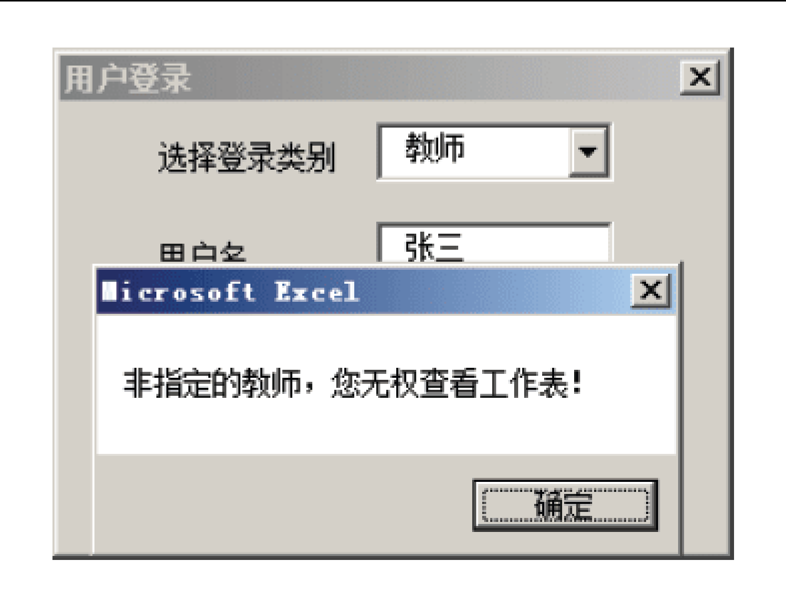



图 20.40　姓名非指定教师时的提示



图 20.41　提示输入姓名或密码

20.5　设置操作权限

不同的操作对象对工作簿具有不同的操作权限，管理员权限最大，能够查看所有工作表并能对程序代码进行修改。教师权限次之，能够使用“学生成绩查询表”和“学生成绩查询结果”工作表。而学生权限最低，只能查看“学生成绩查询结果”工作表。通过只对具有权限的用户才显示特点工作表这种方式，能够方便地实现操作权限的分配。

20.5.1　设定教师权限

学生的操作权限是只能在“学生成绩查询结果”工作表中看到查询的成绩，而其他工作表均不可见。教师的操作权限，是能够使用“学生成绩查询表”查询学生成绩。为了实现这些权限，首先要将需要保护的工作表隐藏起来，只保留所有人都能够看到的“学生成绩查询结果”工作表，然后再对具体的对象显示相应的工作表。

（1）首先实现除“学生成绩查询结果”工作表外的所有工作表的隐藏。为 Workbook 对象的 Open 事件添加事件代码，该事件代码在打开工作簿时运行，将隐藏工作簿中的工作表，同时显示登录窗体，如图 20.42 所示。

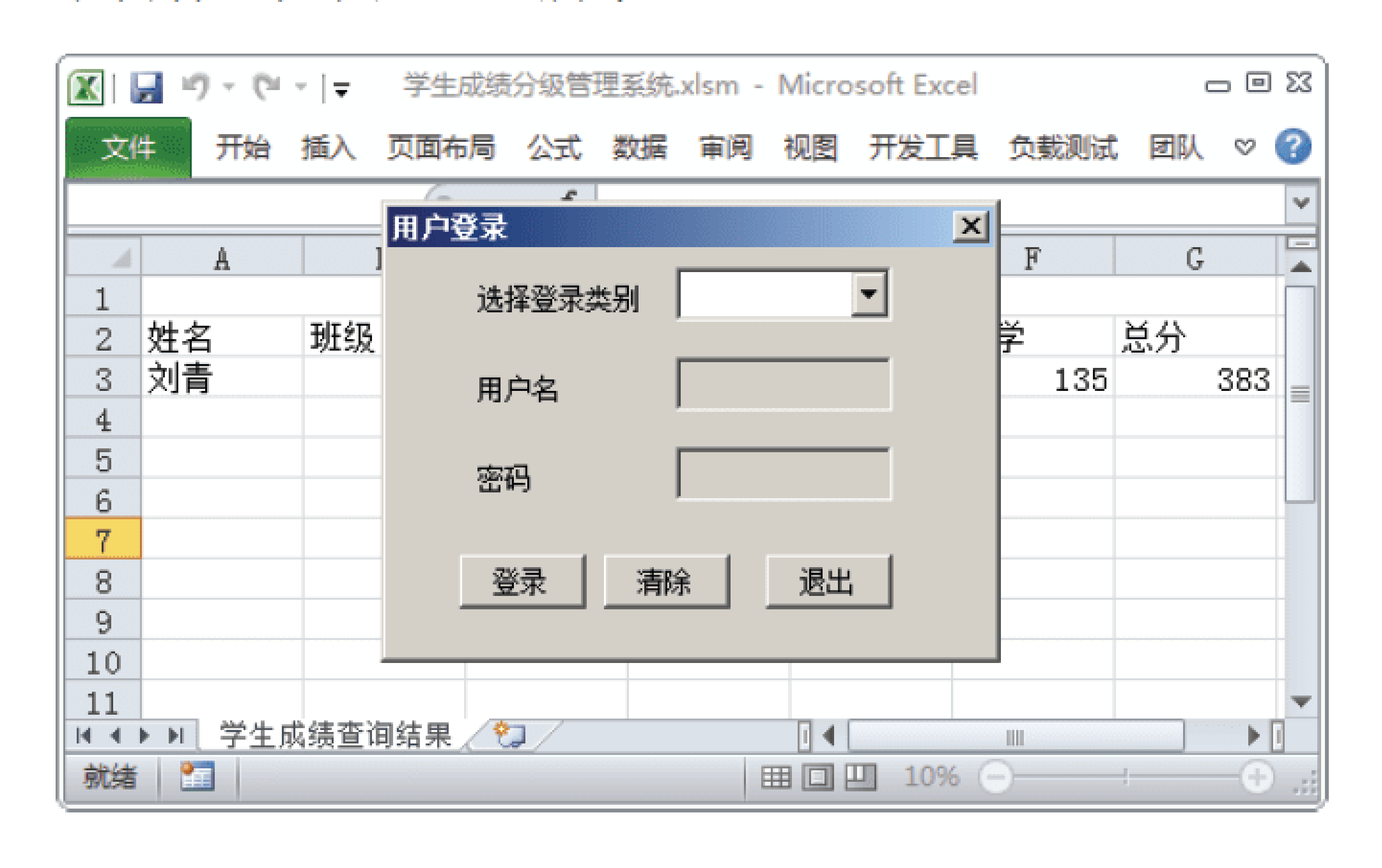


图 20.42　隐藏工作表并显示登录窗口

具体的程序代码如下所示：

```
Private Sub Workbook_Open()
```

```
    Dim n As Integer
    For n = 1 To Sheets.Count                    '遍历所有工作表
        If Sheets(n).Name <> "学生成绩查询结果" _
        Then                                     '如果不是"学生成绩查询结果"工作表
            Sheets(n).Visible = False            '工作表不可见
        End If
    Next
    UserForm1.Show                               '显示用户窗体
End Sub
```

提示：在这段代码中，Sheets.Count 表示工作簿中工作表的数量。使用 For...Next 结构遍历所有的工作表，将工作表 Name 属性非“学生成绩查询结果”的工作表隐藏。

（2）教师能够使用“学生成绩查询表”来查询有关内容，在工作表初始状态为隐藏的情况下，只需要工作表对指定的教师显示即可。在 20.3.2 小节的操作步骤 5 的所创建的代码中，为每一个判断是否是指定老师的 If 结构中添加如下代码，即可实现指定教师使用“学生成绩查询表”。

```
Sheets("学生成绩查询表").Visible = True
```

20.5.2　设置管理员权限

管理员要能查看所有工作表，同时也只有管理员能够打开 VBA 工程并对其中的程序代码进行修改。

（1）程序运行时，当在“选择登录类别”、“组合框”控件中选择了“管理员”选项后，单击“登录”按钮，如果没有输入姓名和密码或姓名密码输入错误时，则程序给出提示，如图 20.43 所示。如果姓名和密码输入正确，则工作簿中所有工作表可见，同时程序显示欢迎对话框，如图 20.44 所示。

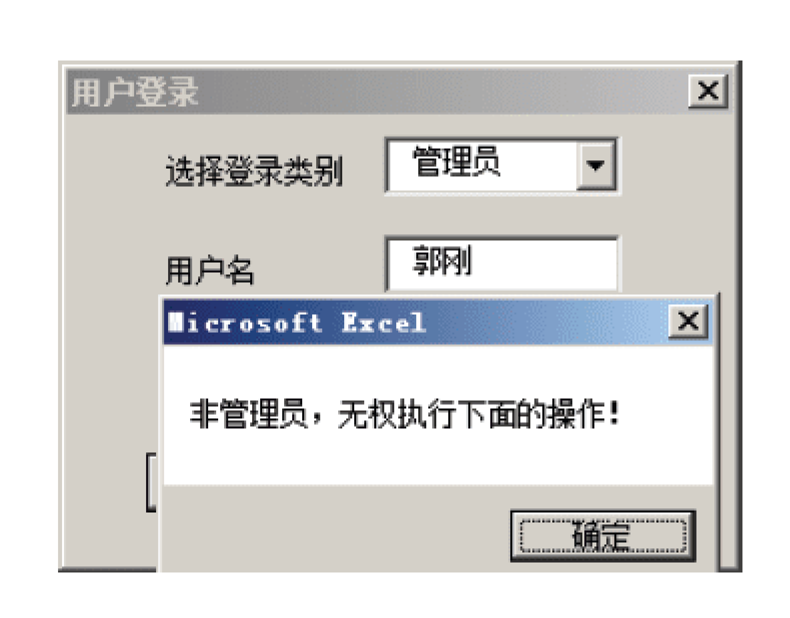


图 20.43　密码输入错误时的提示

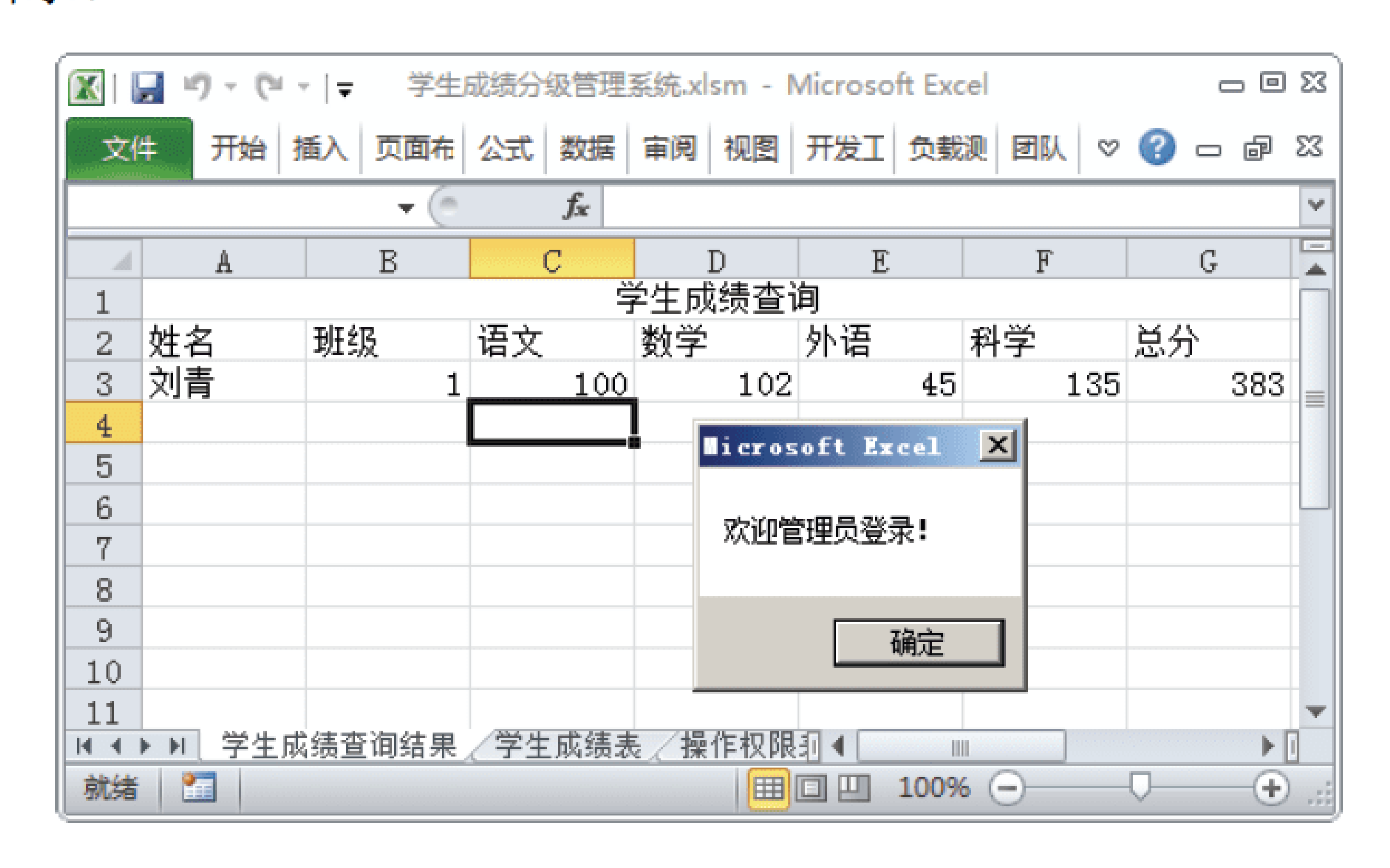


图 20.44　显示欢迎对话框并显示所有工作表

实现上述功能的程序代码放置于“登录”按钮的 Click 事件代码最后，详细的代码如下所示：

```
    If ComboBox1.Value = "管理员" Then                '如果选择管理员
```

```
        If TextBox1.Text <> "" And TextBox2.Text <> "" Then
                                                          '如果输入了姓名和密码
            If TextBox1.Text = "郭刚" And TextBox2.Text = _
             "123456" Then                                '判断是否是管理员
                For n = 1 To Sheets.Count                 '遍历所有工作表
                    Sheets(n).Visible = True              '工作表可见
                Next
                Unload Me                                 '卸载窗体
                MsgBox "欢迎管理员登录！"                  '显示欢迎对话框
                Exit Sub                                  '退出过程
            Else
                MsgBox "非管理员，无权执行下面的操作！"    '提示非管理员
                Exit Sub                                  '退出过程
            End If
        Else
            MsgBox "请输入姓名和密码！"                 '提示输入姓名和密码
            TextBox1.SetFocus                           '"姓名"文本框获得焦点
            Exit Sub                                    '退出过程
        End If
    End If
```

（2）普通用户不应该具有对程序修改的权限，只有管理员具有这样的权限。在工程管理器中选择 VBAProject 选项，然后在弹出的快捷菜单中选择“VBAProject 属性”命令，打开“VBAProject－工程属性”对话框。在对话框的“保护”选项卡中选中“查看时锁定工程”复选框，并设置密码，如这里将其设置为“123456”，如图 20.45 所示。单击“确定”按钮完成设置，工程将会被保护，只有拥有权限密码的用户才能打开工程对程序进行修改，如图 20.46 所示。

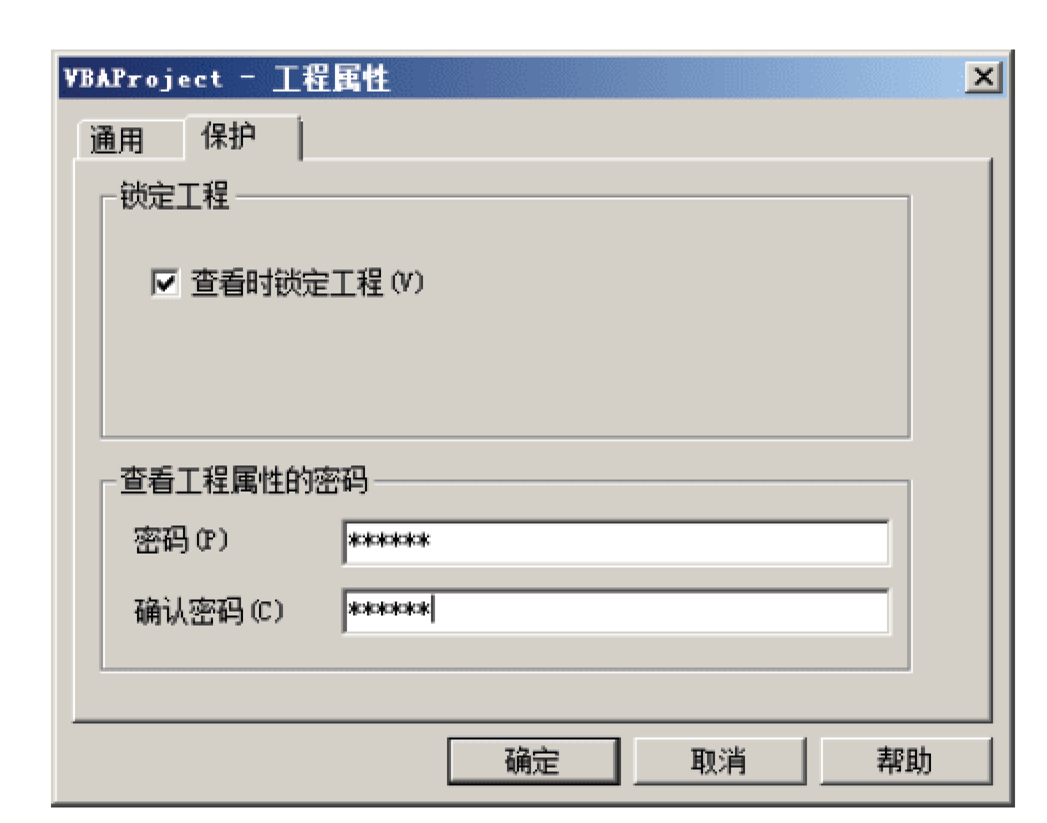


图 20.45　“保护”选项卡中的设置

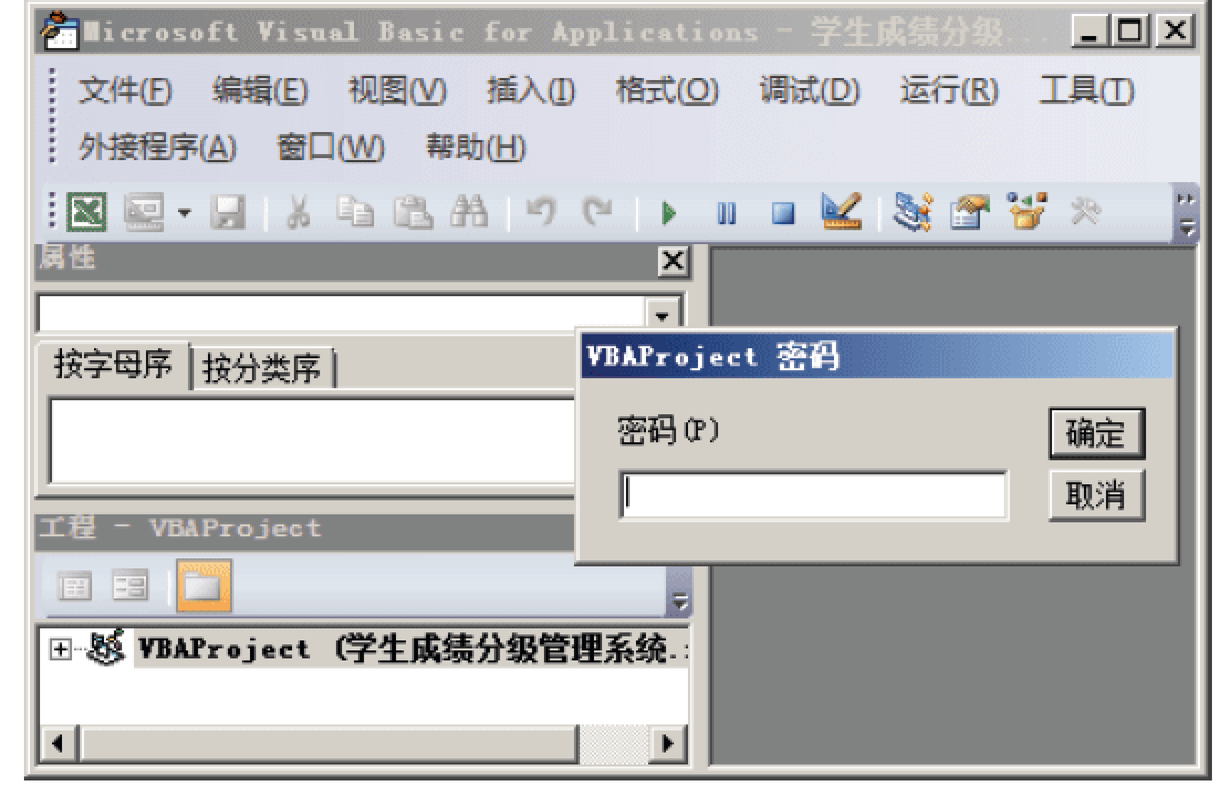


图 20.46　打开工程时需要输入密码

20.6　退 出 程 序

在“登录窗口”中，单击“清除”按钮，将能够清除窗体中所有控件的内容，以便于重新输入登录信息。而单击“退出”按钮，将能够退出 Excel 2010 应用程序。

（1）双击“登录窗口”中的“清除”按钮为按钮添加 Click 事件代码。按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton2 Click()
    Sheet1.Rows(3) = ""                  '查询结果清空
    ComboBox1.Value = ""                 '"组合框"控件恢复初始状态
    TextBox1.Text = ""                   '清空"文本框"控件内容
    TextBox2.Text = ""
    TextBox1.Enabled = False             '"文本框"控件不可用
    TextBox1.BackColor = &H8000000F      '"文本框"控件背景色恢复初始的灰色
    TextBox2.Enabled = False
    TextBox2.BackColor = &H8000000F
End Sub
```

当按钮被单击时，窗体上控件恢复到初始状态，同时清除“学生成绩查询结果”工作表中可能存在的学生查询记录，如图 20.47 所示。

提示： 由于可能学生已经进行了分数查询，因此在按钮事件中，首先将工作表中存放学生查询记录的行清空。

（2）为“退出”按钮添加 Click 事件代码。“退出”按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton3 Click()
    m = MsgBox("真的要退出系统吗？", vbOKCancel)     '提示是否退出系统
     If a = vbCancel Then                          '如果单击“取消”按钮
        Exit Sub                                   '退出过程
   Else
        Workbooks.Close                            '关闭工作簿
   End If
End Sub
```

当单击该按钮时，程序给出提示对话框询问是否退出系统，如图 20.48 所示。单击“确定”按钮，将退出系统；单击“取消”按钮，将关闭对话框，并返回“登录窗口”。

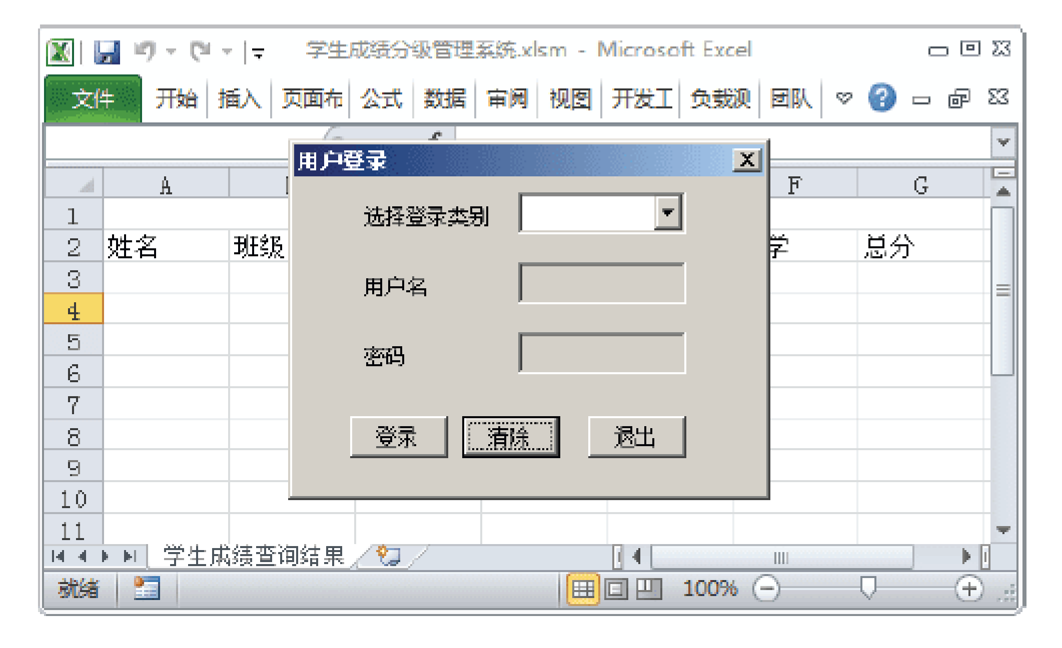


图 20.47　单击“清除”按钮，控件恢复初始状态



图 20.48　退出系统时的提示对话框

20.7　小　　结

通过本章的学习，读者将能够进一步理解 VBA 编程的一般思路和方法。同时，读者将能够掌握使用 VBA 设计用户登录界面的方法，掌握为不同用户设计不同操作权限的方法和技巧。通过本章实例的制作，读者应该注意下面这些知识要点：

- 本实例使用 Match 函数来实现根据单一条件来获取记录，同时，使用 AdvancedFilter 方法来实现多条件查询。数据的查询是对管理数据表的常用操作，这里介绍的方

法实用而有效。

- 设计用户操作界面离不开用户窗体和窗体中的控件，对控件的编程就是设置控件属性和使用控件方法的过程。例如，“文本框”控件的 PasswordChar 属性的使用，“组合框”控件的 Value 属性的作用，这些属性在本例中都是编程的关键。
- 许多应用程序都需要登录界面，对用户登录的分级管理是实现工作表的保护、分配操作权限的有效措施。本章实例提供了实现登录的密码功能和创建登录用户的分级管理的一种有效方法。

第 21 章　档案管理系统

档案管理系统是一个对人员信息管理的系统，用于保存人员的个人档案信息。系统中提供了档案信息的录入界面，并提供了对信息进行添加、修改和查询的功能，并实现退出程序和查看工作表的功能。本章学习的内容如下：

- ❑ 掌握档案管理系统的需求分析方法；
- ❑ 掌握档案管理系统的用户界面设计方法；
- ❑ 掌握对工作表数据进行查询、录入和修改等操作的实现方法。

21.1　设 计 功 能

本章制作的系统是档案管理系统，该系统能够实现对 Excel 人事工作表中的人事记录进行添加、修改和查询。功能的实现主要通过对用户窗体的控件和工作表的操作来实现。

21.1.1　功能简介

这是一个教师人事管理系统，系统能够实现对 Excel 工作表中的人事记录信息进行各种操作。启动 Excel 2010，打开工作簿后，自动打开“教师人事管理系统”用户窗体，如图 21.1 所示。



图 21.1　程序运行界面

本实例要求在窗体中输入员工人事信息记录的各个项目，单击“添加”按钮将输入的记录添加到工作表中。此时，如果未输入职工编号、姓名或输入人员记录在工作表中已经存在，系统会给出提示。

程序对人员的人事资料可以按照输入的姓名来进行查询。在“姓名”文本框中输入需要查询人员的姓名，单击“查询”按钮，找到的人员信息将会在用户窗体的各个控件中显示。在获得查询结果后，可以对窗体中显示的某个职工的信息资料进行修改。完成修改后，单击“修改”按钮，能够将新的记录重新写入工作表中。

实例在打开工作簿后，为了保护工作表，人事信息工作表默认状态下是隐藏的，具有操作权限的用户可查看工作表的内容。即单击窗口中的“查看”按钮，系统要求用户输入查看密码，具有权限的用户在输入密码后能够查看完整的人事信息工作表。

21.1.2　设计思路

实例是基于 Excel 工作表的应用程序，在工作表中保存人事资料，使用窗体来构建用户操作界面。在本实例中，对人事信息数据的操作是通过用户窗体中控件的操作来完成。窗体中主要使用了“文本框”控件、“单选按钮”控件、“组合框”控件和“命令按钮”控件，通过对控件进行编程来实现实例需要的功能。本实例根据实现功能的需要，对功能模块进行了划分，如图 21.2 所示。

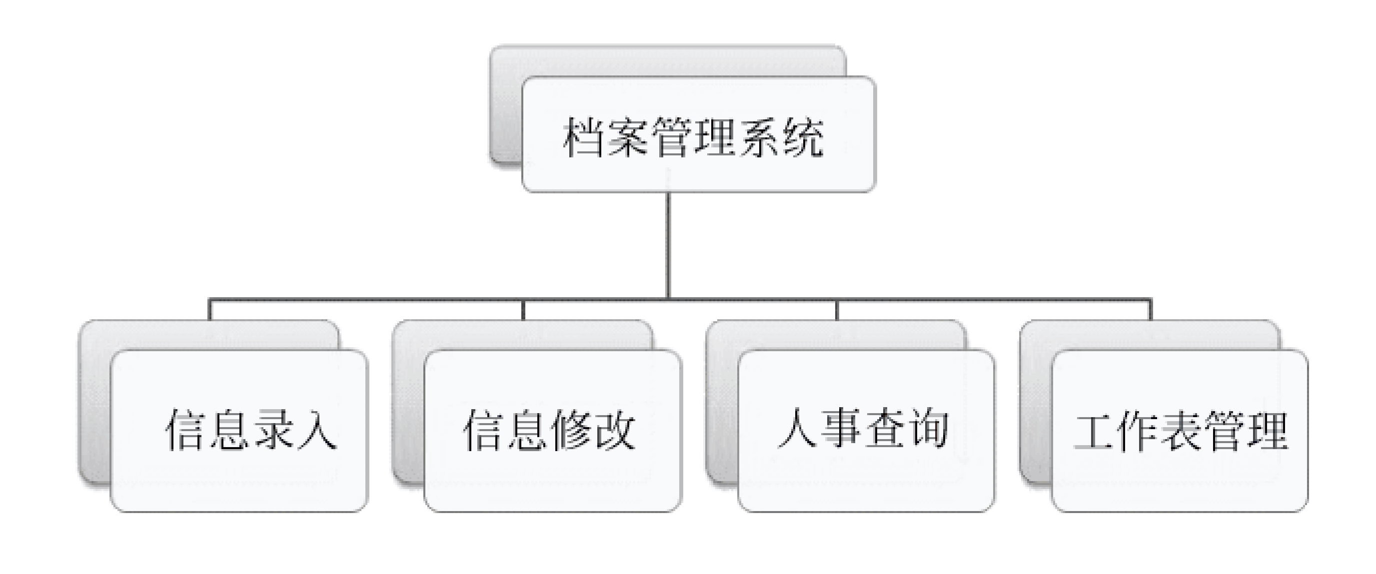


图 21.2　本章实例功能模块图

21.2　设计用户界面

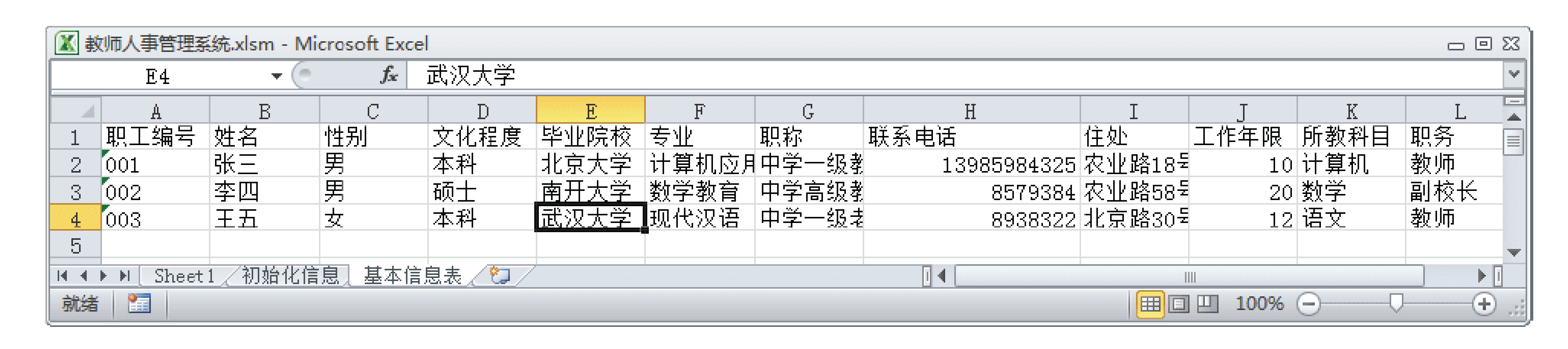
在本例中，人事资料放置于一个 Excel 工作表中，使用用户窗体作为对人事资料进行管理的操作界面，因此需要首先设计操作界面。这主要包括为窗体添加控件、设计窗体中控件的布局以及设置控件的属性等。

21.2.1　添加信息录入控件

这里，首先在窗体中添加用于显示人事资料信息的各类控件，这些控件包括 3 个“组合框”控件、7 个“文本框”控件和 10 个“标签”控件。这里，使用 1 个“框架”控件将这些控件合为一组，使它们与窗体中的“按钮”控件分隔开。添加控件后，对控件的 Caption

属性进行设置，同时调整控件在窗体中的大小和位置。

（1）启动 Excel 2010，新建工作簿。在工作簿中保留 2 个工作表，其中第一个工作表“Sheet1”为一个空白工作表。第二个工作表命名为“基本信息表”，其结构如图 21.3 所示。



职工编号	姓名	性别	文化程度	毕业院校	专业	职称	联系电话	住处	工作年限	所教科目	职务
001	张三	男	本科	北京大学	计算机应	中学一级	13985984325	农业路18	10	计算机	教师
002	李四	男	硕士	南开大学	数学教育	中学高级	8579384	农业路58	20	数学	副校长
003	王五	女	本科	武汉大学	现代汉语	中学一级	8938322	北京路30	12	语文	教师

图 21.3　创建 Excel 工作表

（2）打开 Visual Basic 编辑器。在工程资源管理器中添加一个用户窗体，设置窗体的属性。这里，主要是更改窗体的 Caption 属性和设置窗口的宽度和高度，如图 21.4 所示。

（3）从“控件工具箱”中向窗体中选择“框架”控件。在用户窗体中拖动光标绘制该控件，右击该控件，在弹出的快捷菜单中选择“属性”命令，打开“属性”对话框，设置控件标题，如图 21.5 所示。

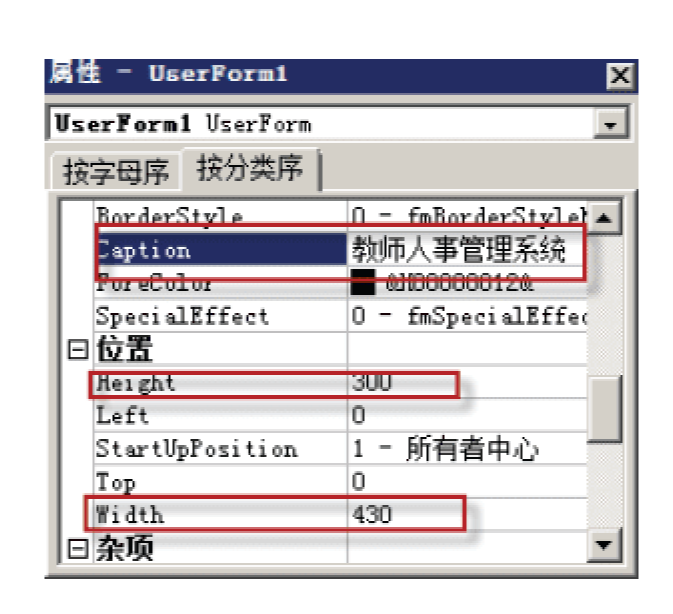


图 21.4　设置窗体属性

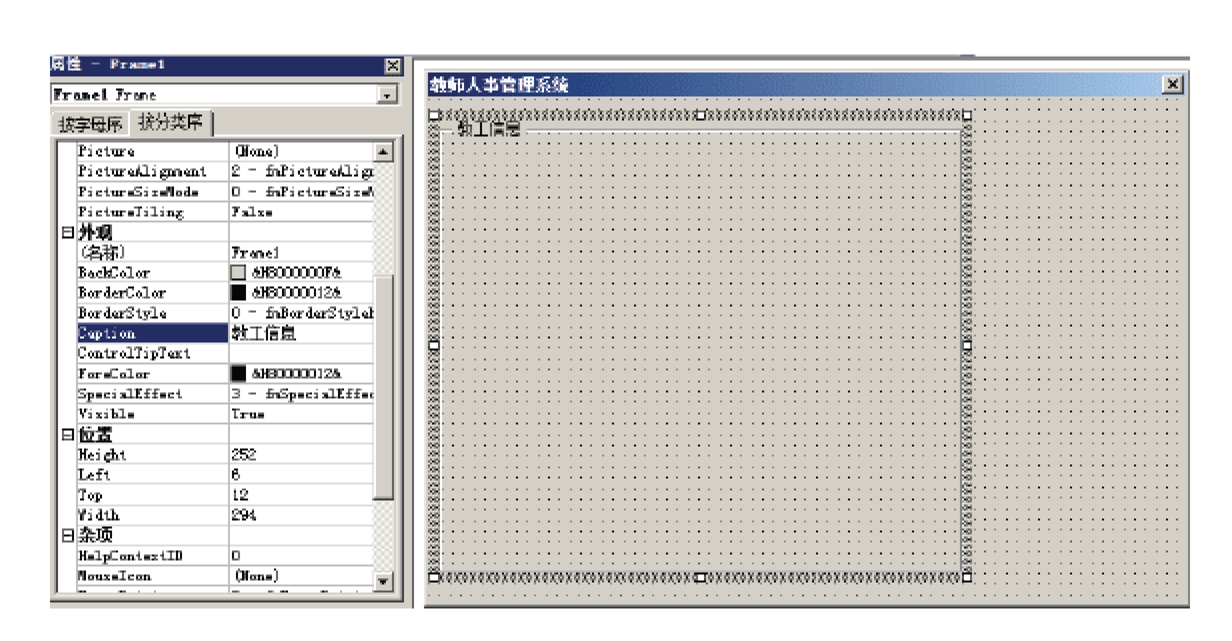


图 21.5　添加“框架”控件，并设置其属性

（4）“框架”控件划分了用于显示人事信息资料的控件的放置区域，在这个区域内再添加一个“框架”控件，将标题更改为“性别”。在该控件内放置 2 个“单选按钮”控件，将控件标题更改为“男”和“女”。拖动控件边框，调整“单选按钮”控件在“框架”控件中的位置和大小，效果如图 21.6 所示。

（5）向窗体中添加 2 个“标签”控件和 2 个“文本框”控件。“标签”控件的标题分别更改为“员工编号”和“姓名”。适当调整控件的大小，将“标签”控件和“文本框”控件搭配放置于框架的顶端，如图 21.7 所示。

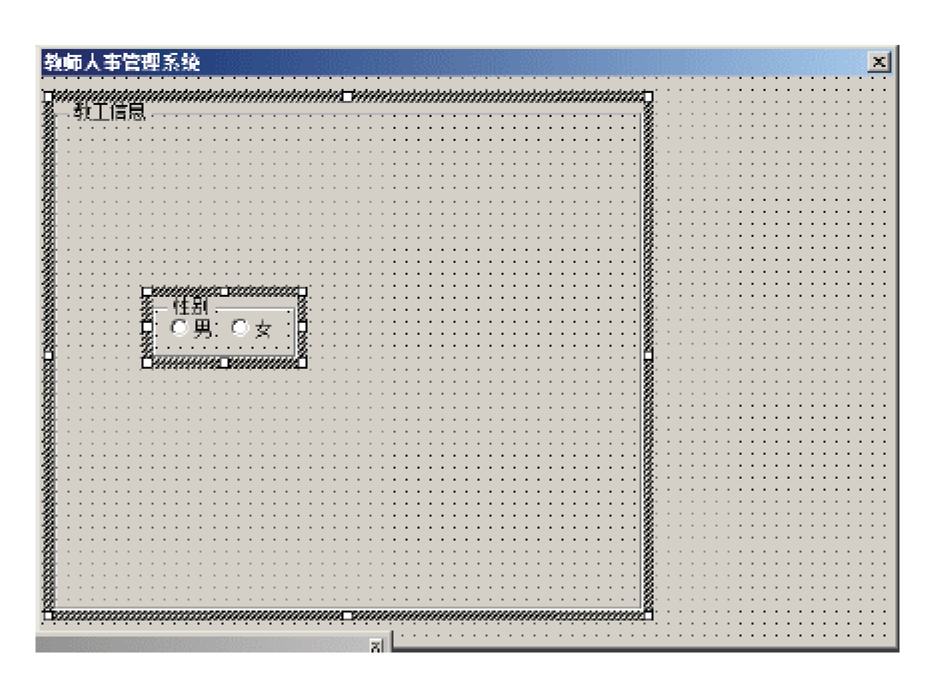


图 21.6　添加“框架”控件和“单选按钮”控件

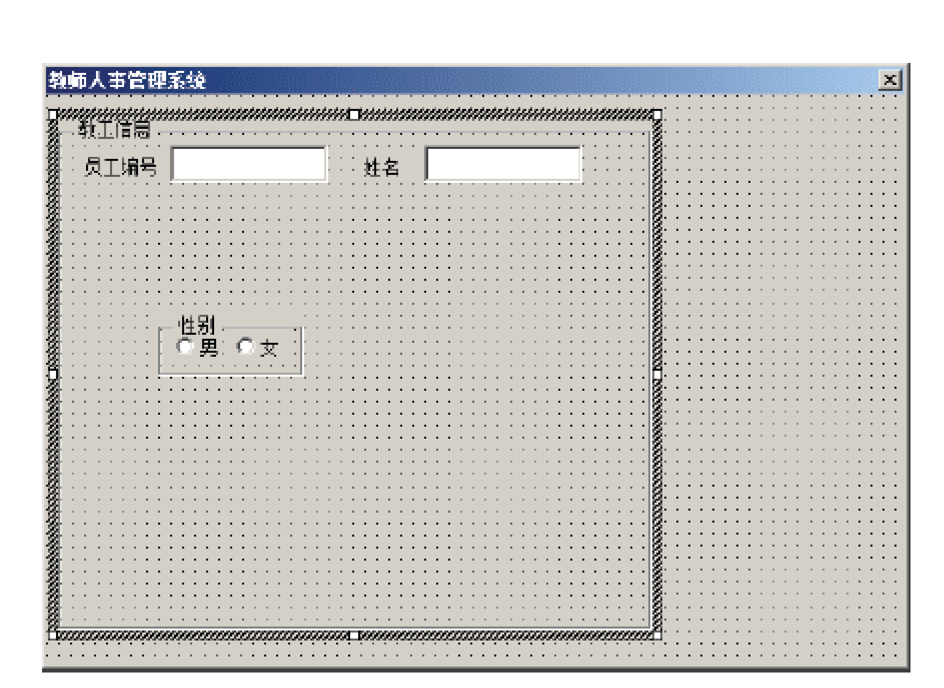


图 21.7　控件放置在框架的顶端

（6）按住 Ctrl 键，同时选择“文本框”控件和“标签”控件后并右击，在弹出的快捷菜单中选择“对齐”→“顶端对齐”命令，使控件顶端对齐，如图 21.8 所示。

（7）拖动“框架”控件，将控件放置于“员工编号”、“标签”控件的下方适当位置。选择“员工编号”、“标签控件”和“框架”控件后右击，在弹出的快捷菜单中选择 “对齐”→“左对齐”命令，使控件左对齐，如图 21.9 所示。

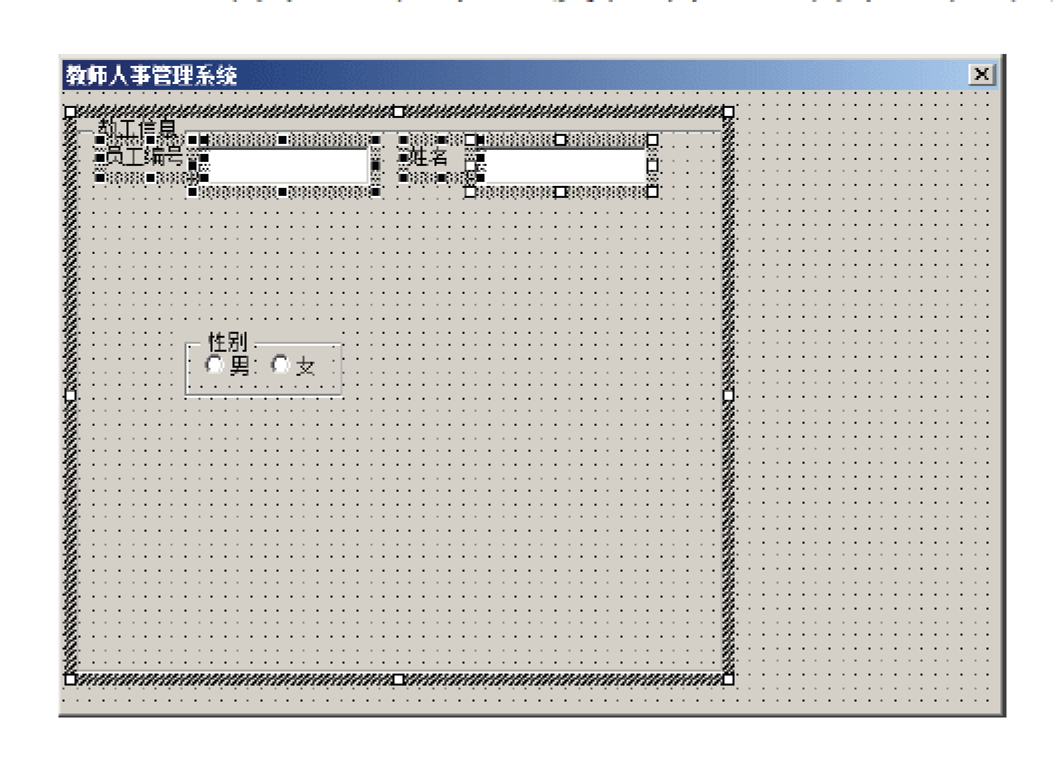


图 21.8　放置控件

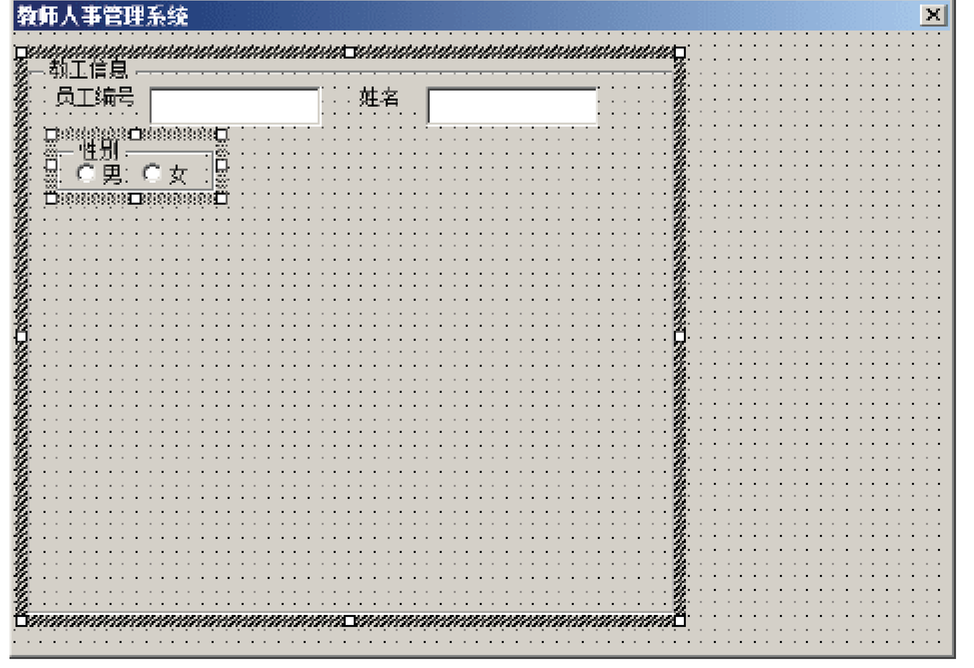


图 21.9　控件左对齐

注意：将控件放置于“框架”控件后，其与“框架”控件形成了一种绑定关系，当改变“框架”控件的位置时，其内部的控件能够自动跟随其改变位置。

（8）向“教工信息”框架中添加“标签”控件、“组合框”控件和“文本框”控件。这里“标签”控件作为“文本框”控件或“组合框”控件的标题，设置“标签”控件的 Caption 属性，采用上面的介绍，设置控件间的相对位置。完成设置后，控件在窗体中的布局如图 21.10 所示。

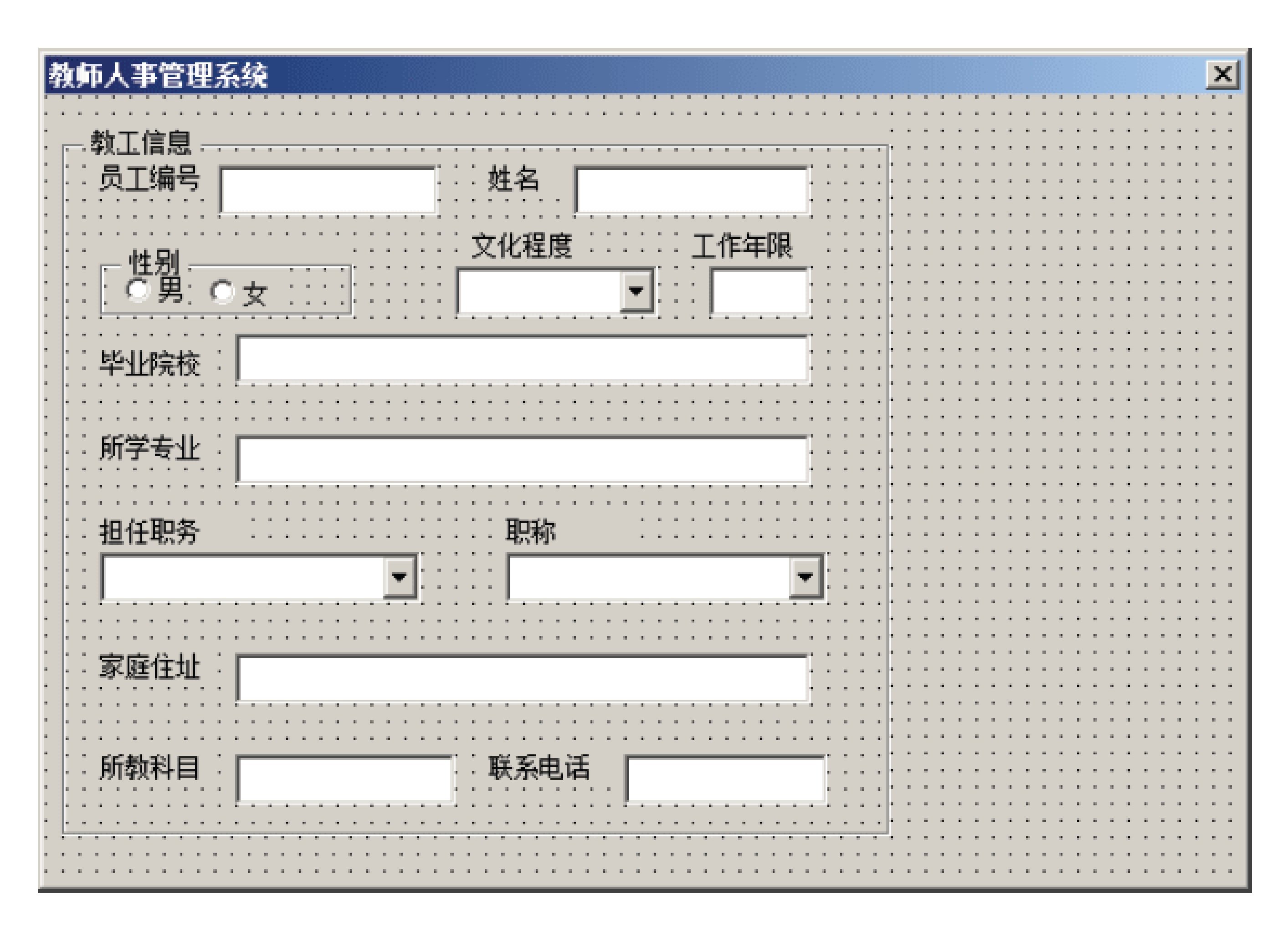


图 21.10　完成设置后的控件布局

21.2.2　添加控制按钮控件

这里一共需要 5 个“命令按钮”按钮控件来实现对程序的控制，这些按钮放置于用户

窗体的右侧。按钮添加后，除了需要修改显示的名称和放置它们之外，还需要设置它们的 ControlTipText 属性。

（1）在用户窗体的右侧放置 5 个“按钮”控件，分别对这 5 个“按钮”控件的属性进行设置。设置属性时，首先设置控件的 Caption 属性，更改控件的标题。然后设置控件的 Height 属性和 Width 属性更改控件的大小。这里，所有“按钮”控件的这两个属性值设置应相同，这样可以保证控件大小完全相同，如图 21.11 所示。

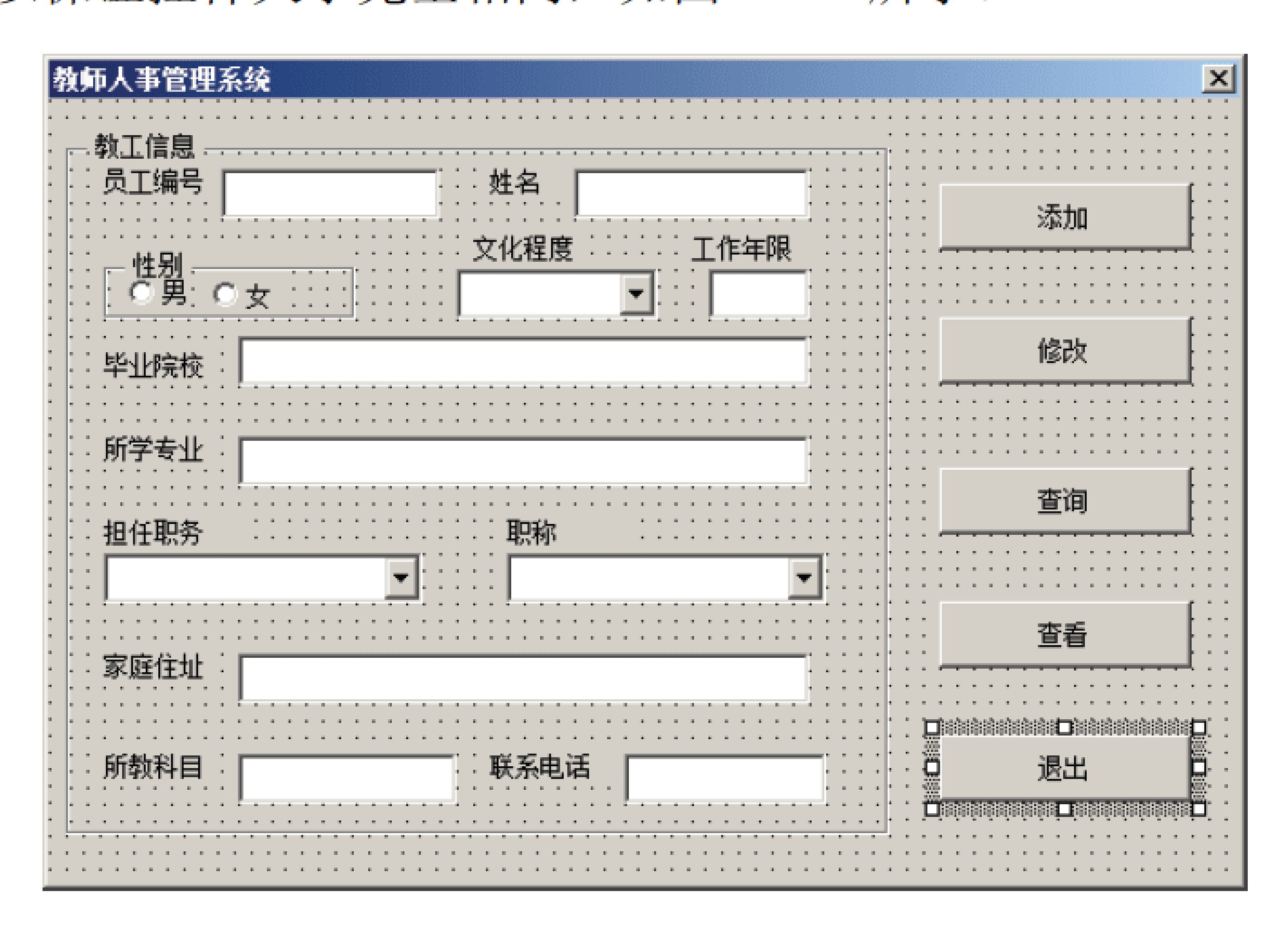


图 21.11　设置控件的属性

提示： 在调整设计用户界面时常常需要调整控件大小，使控件在大小和形状上完全相同。如果通过拖动控件控制柄手动调整，无法使控件大小精确统一。此时，可以像本步使用的方法那样将控件的 Width 和 Heigth 设置的相同来实现对大小的精确调整。

（2）单击“退出”按钮，在“属性”对话框中设置控件的 ControlTipText 属性，如图 21.12 所示。设置该属性后，程序运行时，当光标放置于按钮上时，程序会自动给出输入的提示信息，表明该按钮的作用，如图 21.13 所示。

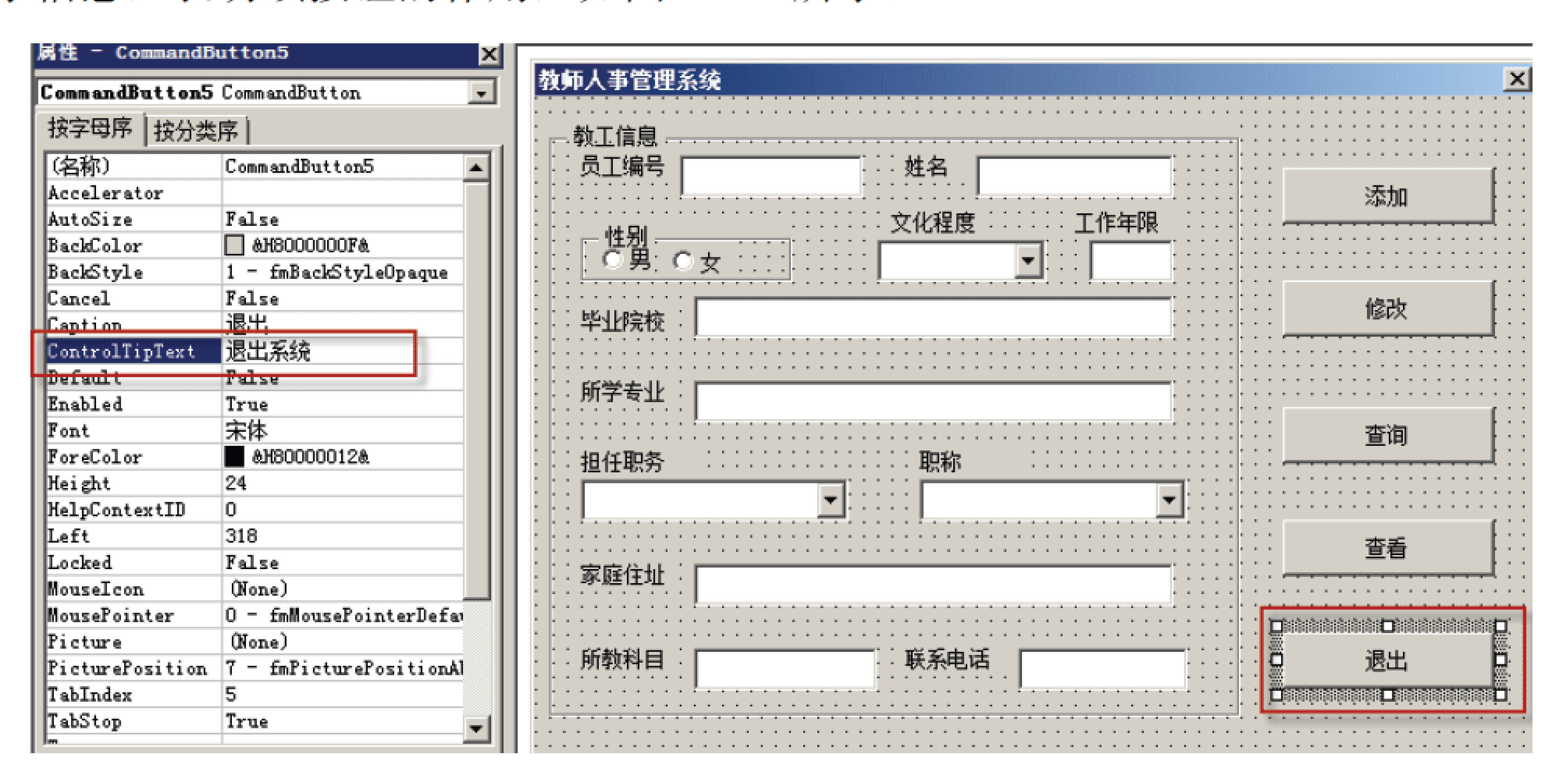


图 21.12　设置控件的 ControlTipText 属性

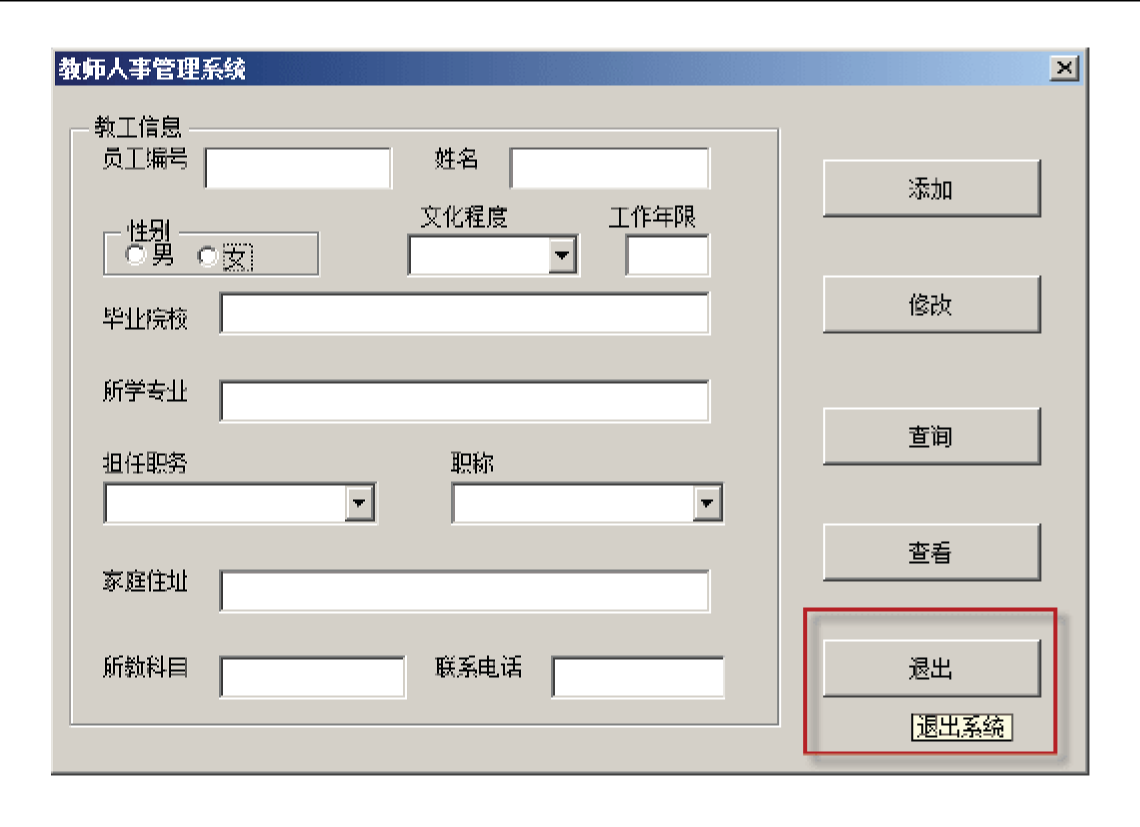


图 21.13　按钮上的提示信息

（3）按住 Ctrl 键，同时选择这 5 个控件并右击，在弹出的快捷菜单中选择“对齐”→“左对齐”命令使控件左对齐，如图 21.14 所示。

提示：在选择一个控件后，按住 Shift 键，单击另一个控件，在这两个控件之间的所有控件将能够同时被选择。

（4）将“退出”按钮和“添加”按钮分别向下和向上拖放到适当的位置，同时选择所有的“命令按钮”控件。选择“格式”→“垂直间距”→“相同”命令使控件等间距排列，如图 21.15 所示。

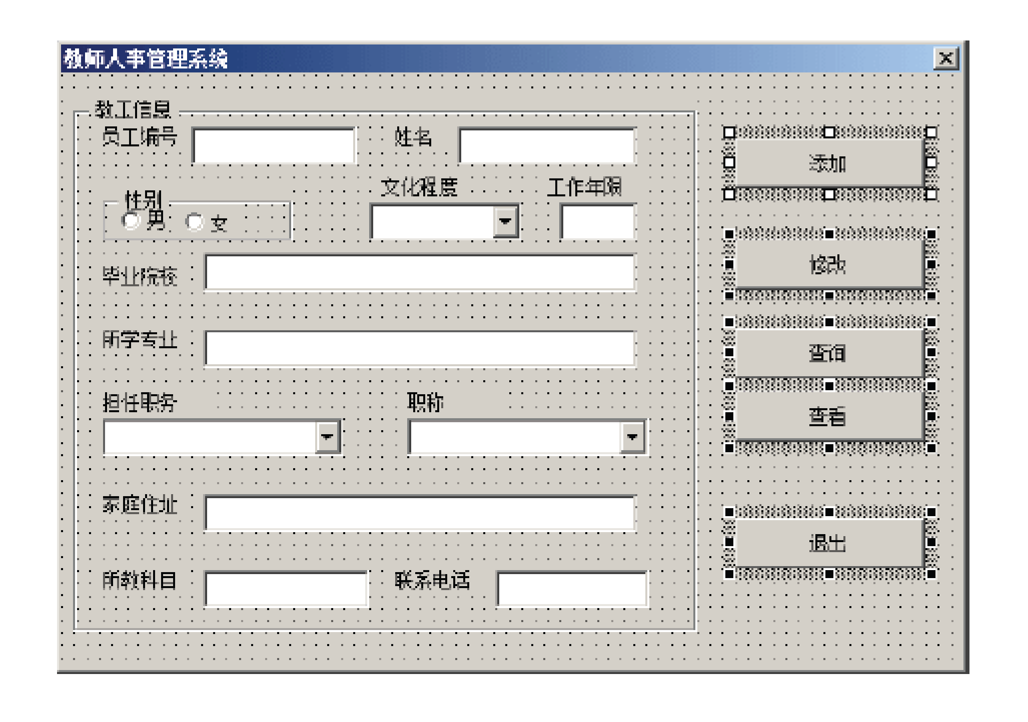


图 21.14　控件左对齐

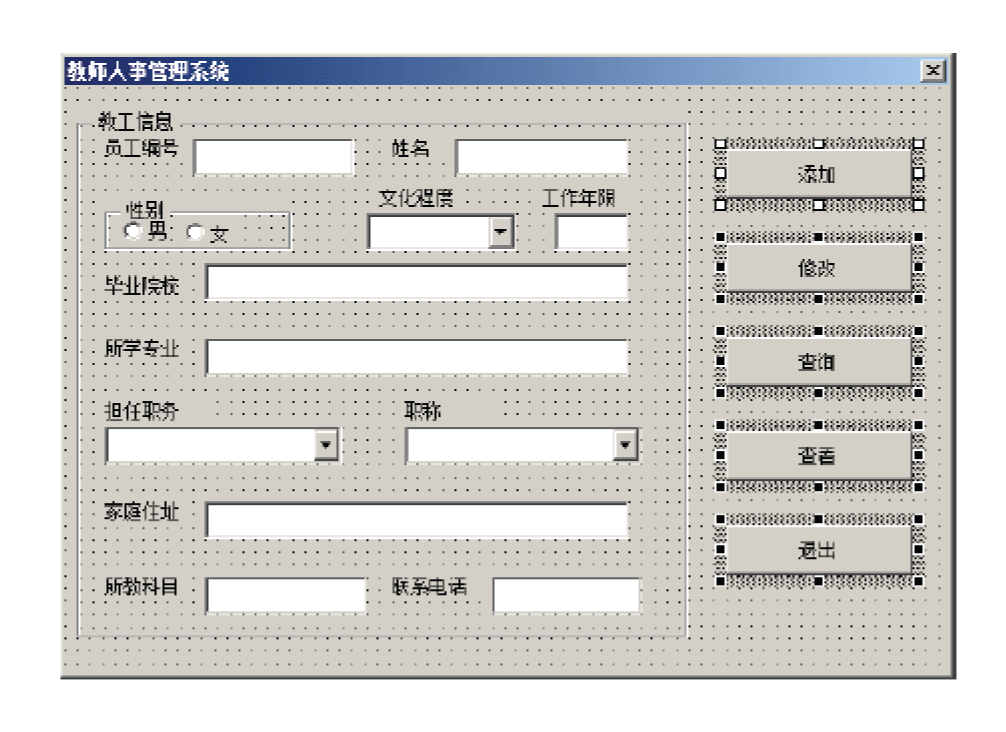


图 21.15　使控件等间距排列

（5）至此，本实例的用户界面设计完成，按 F5 键运行程序，可以查看操作界面中控件的布局效果，如图 21.16 所示。

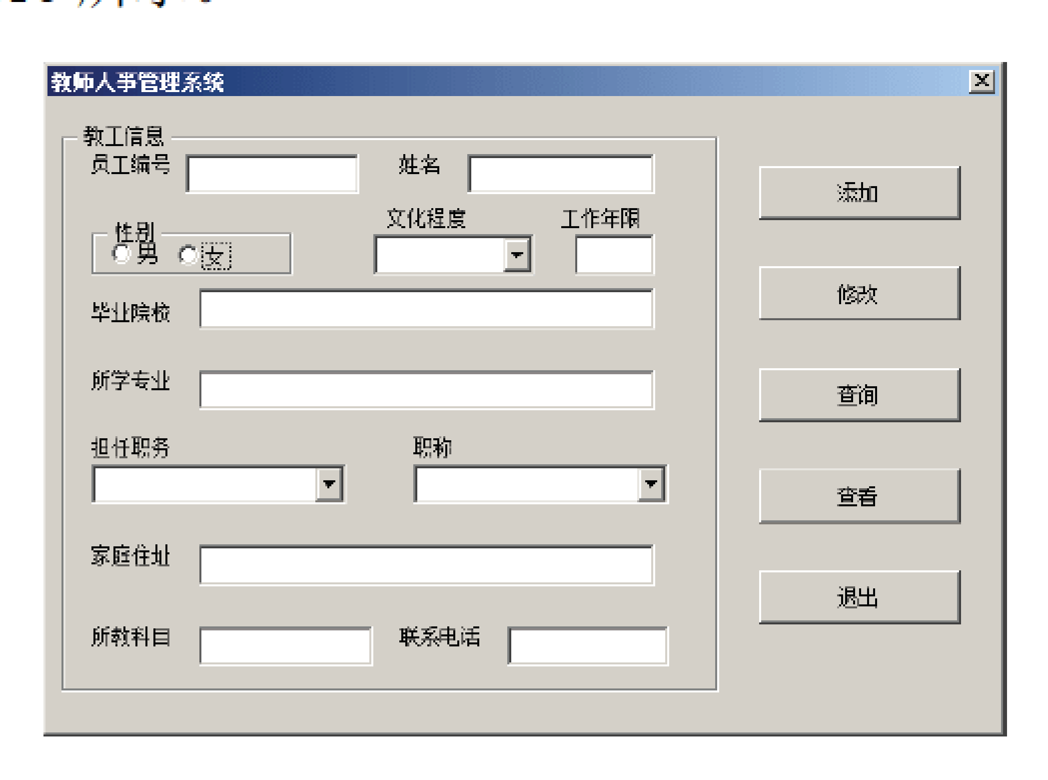


图 21.16　设计完成后的用户操作界面

21.3　实现程序功能

程序代码的编写是实现本实例功能的关键，作为事件驱动的应用程序，编程的关键在于编写各个控件的事件代码。下面依次介绍实例各个功能的实现方法。

21.3.1　实现界面初始化功能

这里，窗体初始化过程主要是通过事件代码向“组合框”控件添加选项。本实例的窗体中共有 3 个“组合框”控件，选项内容不同，但是操作代码基本相同。因此将操作代码放置于一个单独的过程中，控件初始化时只需要调用该过程向控件中依次添加选项即可。将“组合框”控件各个选项的内容放置于一个单独的 Excel 工作表中，这样能够方便以后系统的维护，如，当需要增加新的选项时，只要在对该工作表进行修改即可。下面介绍具体的制作步骤：

（1）在工作簿中插入一个名为“初始化信息”的工作表，该工作表为用户界面中的“组合框”控件提供选项。其中，A 列列出了“文化程度”选项的内容，B 列列出“职称”选项的内容，C 列列出“担任职务”选项的内容，如图 21.17 所示。

（2）切换到 Visual Basic 编辑器。在工程资源管理器中添加一个模块“模块 1”，为模块添加一个过程。该过程用于向窗体中的“组合框”控件添加项目，过程的详细代码如下所示：

```
01  Sub AddComboItem(combName As Object, strN As String)
02      Dim rngW As Range                                        '定义对象变量
03      Dim n As Integer
04      Set rngW = Sheets("初始化信息").Rows(1).Find(strN)      '查找指定的列
05      Do While rngW.Offset(n, 0) <> ""                   '如果单元格内容不为空
06          n = n + 1
07          combName.AddItem rngW.Offset(n, 0)             '向控件中添加选项
08      Loop
09     End Sub
```

说明：本段过程包含两个参数，combName 在过程调用时，用于传递“组合框”控件的名称，strN 参数用于指定项目标题。第 04 行代码用于在工作表中查找项目标题，找到项目后使用 Do While...Loop 结构遍历该列的所有单元格。这里，使用 Offset 属性，每次循环向下偏移一个单元格，然后将该单元格的内容添加到组合框中。

（3）双击添加的用户窗体，打开用户窗体的“代码”窗口，为窗体添加 Initialize 事件代码。添加初始化代码后，程序的运行效果如图 21.18 所示。该段代码的作用是在窗体载入时对窗体中的控件初始化。

```
Private Sub UserForm Initialize()
    AddComboItem ComboBox1, "文化程度"                    '调用过程添加项目
    AddComboItem ComboBox2, "担任职务"
    AddComboItem ComboBox3, "职称"
End Sub
```

说明： 这里调用上一步创建的过程向窗体中的“组合框”控件添加项目。调用过程时需要传递两个参数，一个是“组合框”控件名，一个是“初始化信息表”的列标题。

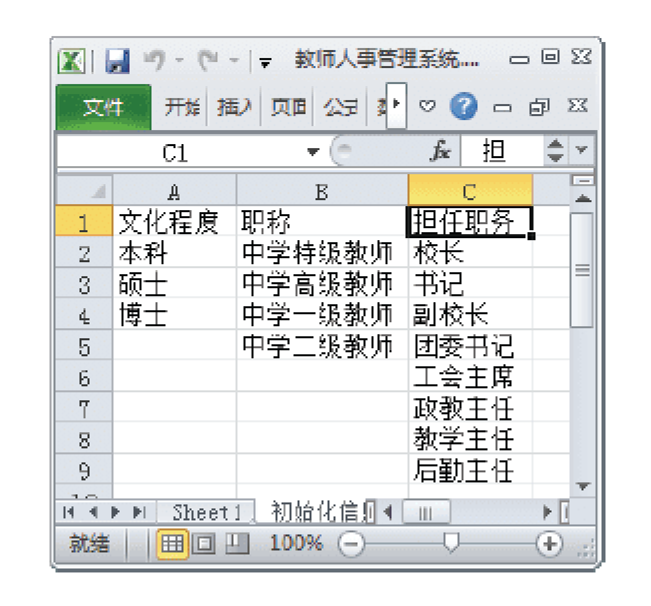


图 21.17　创建“初始化信息”表

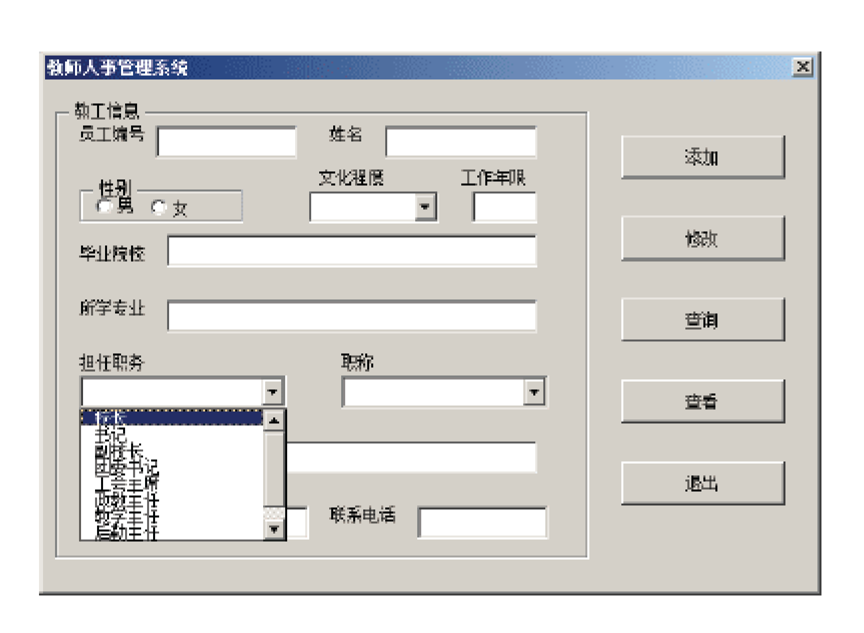


图 21.18　窗体初始化效果

21.3.2　实现添加人事信息功能

程序运行时，在窗体中输入职工的各项人事信息，单击“添加”按钮，将信息添加到工作表中。添加信息首先要对输入情况进行判断，以避免输入错误。这里，以“姓名”和“职工编号”作为判断的依据，查询工作表中是否有相同的姓名或职工编号，如果有，则说明信息重复，需要重新输入。

另外，程序还需要判断信息输入是否完整。这主要包括“姓名”和“编号”是否输入，“工作年限”和“电话号码”是否是数字。

在完成对输入信息输入是否正确的判断后，即可将窗体控件的内容逐一写入对应的工作表单元格中。这里，将信息写入单元格的程序代码放置于一个独立的过程中，在使用时直接调用即可。完成信息输入后，清除窗体中控件内容，以便下一次的输入。本节程序流程图如图 21.19 所示。下面介绍具体的制作步骤。

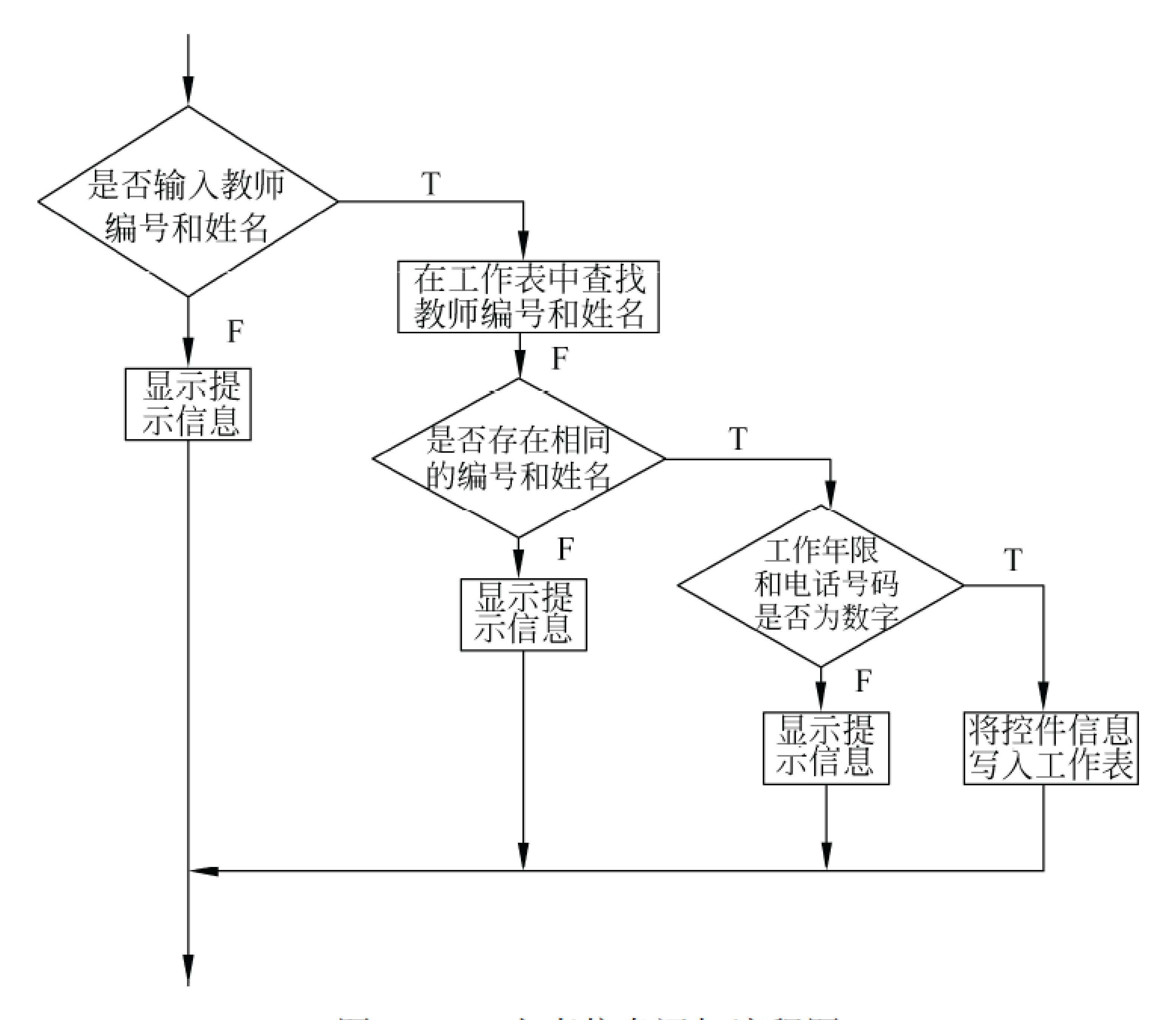


图 21.19　人事信息添加流程图

（1）在“模块 1”中添加一个 UpdateD 过程，该过程用于更新“基本信息”工作表中的教师的信息资料，其代码如下：

```
Sub UpdateD(ws As Worksheet, r As Integer)
    With ws
        .Cells(r, 1) = UserForm1.TextBox1.Text              '写入编号
        .Cells(r, 2) = UserForm1.TextBox2.Text              '写入姓名
        .Cells(r, 10) = UserForm1.TextBox3.Text             '写入工作年限
        .Cells(r, 5) = UserForm1.TextBox4.Text              '写入毕业学校
        .Cells(r, 6) = UserForm1.TextBox5.Text              '写入所学专业
        .Cells(r, 9) = UserForm1.TextBox6.Text              '写入家庭住址
        .Cells(r, 11) = UserForm1.TextBox7.Text             '写入所教科目
        .Cells(r, 8) = UserForm1.TextBox8.Text              '写入联系电话
        If UserForm1.OptionButton1.Value = True _
        Then .Cells(r, 3) = "男"                            '性别写入“男”
        If UserForm1.OptionButton2.Value = True _
        Then .Cells(r, 3) = "女"                            '性别写入“女”
        .Cells(r, 4) = UserForm1.ComboBox1.Value            '写入文化程度
        .Cells(r, 12) = UserForm1.ComboBox2.Value           '写入担任职务
        .Cells(r, 7) = UserForm1.ComboBox3.Value            '写入职称
    End With
End Sub
```

提示：这里，过程中的参数 ws 用于在调用该过程时指定需要写入数据的工作表，参数 r 用于确定需要写入数据的单元格的行号。

（2）在“模块 1”中创建一个名为“ClearAll”的过程，该过程用于将窗体中控件内容清空，其代码如下所示：

```
Sub ClearAll()
    Dim c As Control                             '定义对象变量
    For Each c In UserForm1.Controls             '遍历窗体中所有控件
        If TypeName(c) = "TextBox" Then c.Text = ""
                                                 '“文本框”控件内容清空
        If TypeName(c) = "ComboBox" Then c.Text = ""
                                                 '“组合框”控件内容清空
        If TypeName(c) = "OptionButton" Then c.Value = 0
                                                 '“单选按钮”控件复位
    Next
End Sub
```

提示：这里，使用 For Each...In Next 结构遍历窗体中所有控件，对这些控件中的“文本框”控件、“组合框”控件和“单选按钮”控件进行初始化处理。使用 TypeName 函数获取控件名称，使用该名称作为判断控件类型的标志。

（3）为窗体中的“添加”按钮添加 Click 事件代码。该事件代码调用 Update 过程向工作表中写入输入的职工信息，同时判断输入的情况。“添加”按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton1 Click()
    Dim n As Integer
    Dim rng As Range, rng1 As Range
```

```
    If UserForm1.TextBox1.Text = "" Then            '判断是否输入职工编号
        MsgBox "未输入职工编号，请输入！"              '给出提示
        UserForm1.TextBox1.SetFocus                 '文本框获得焦点
        Exit Sub                                    '退出过程
    End If
    If UserForm1.TextBox2.Text = "" Then            '判断是否输入姓名
        MsgBox "未输入职工姓名，请输入！"              '给出提示
        Userform2.TextBox2.SetFocus                 '文本框获得焦点
        Exit Sub                                    '退出过程
    End If
    Set rng = Sheets("基本信息表").Cells.Find(TextBox2.Text)
                                                    '工作表中查找相同姓名
    Set rng1 = Sheets("基本信息表").Cells.Find(TextBox1.Text)
                                                    '查找相同职工编号
    If Not rng Is Nothing Then                      '如果存在同名
        MsgBox "该教师已经存在，请重新输入！"          '给出提示
        UserForm1.TextBox2.SetFocus                 '文本框获得焦点
        Exit Sub                                    '退出过程
    End If
    If Not rng1 Is Nothing Then                     '如果编号存在
        MsgBox "职工编号重复，请重新输入"              '给出提示
        UserForm1.TextBox1.SetFocus                 '文本框获得焦点
        Exit Sub
    End If
    If Not IsNumeric(TextBox3.Text) Then            '工作年限是否为数字
        MsgBox "工作年限需输入数字！"                  '不是数字给出提示
        UserForm1.TextBox2.SetFocus                 '文本框获得焦点
        Exit Sub                                    '结束过程
    End If
    If Not IsNumeric(TextBox8.Text) Then            '电话号码是否为数字
        MsgBox "电话号码输入数字！"                    '不是数字给出提示
        UserForm1.TextBox2.SetFocus                 '文本框获得焦点
        Exit Sub                                    '结束过程
    End If
    n = Sheets("基本信息表").Range("a1048576").End(xlUp).Row + 1
                                                    '指向最后一行
    UpdateD Sheets("基本信息表"), n                  '写入信息
    MsgBox ("新信息添加成功！")                        '提示写入成功
    ClearAll                                        '控件初始化
    End Sub
```

提示： 在这段代码中，依次使用 If 结构来判断各项输入是否正确，并进行相应的处理。在完成数据合法性的检验后，调用 UpdateD 过程来向工作表中写入输入的信息，使用该过程需要传递工作表名和单元格行号这两个参数。完成数据写入后，调用 ClearAll 过程来将窗体中的控件初始化，为下次输入做准备。

（4）代码对 3 种输入错误进行处理，当职工编号、职工姓名没有输入以及工作年限和电话号码输入非数字时，程序均给出相应的提示，如图 21.20 所示。职工编号和姓名的输入不允许与工作表中已有数据重复，程序会对其进行查询，如果在工作表中查询到这两个输入项目，则给出提示，如图 21.21 所示。

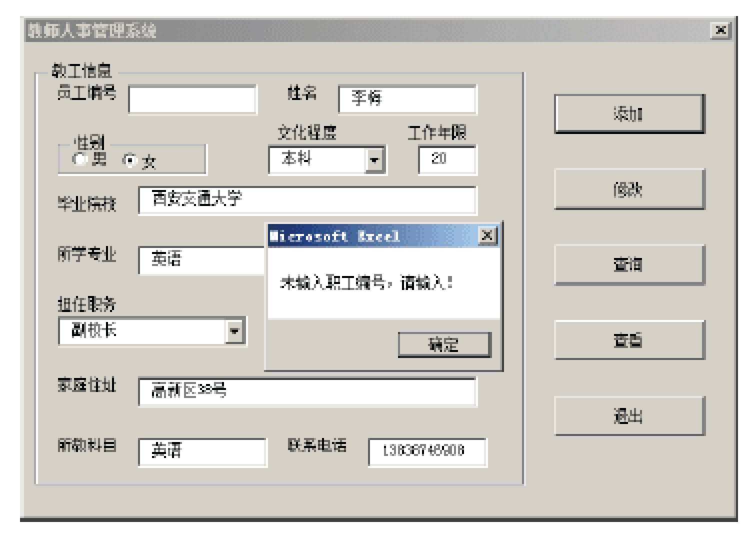
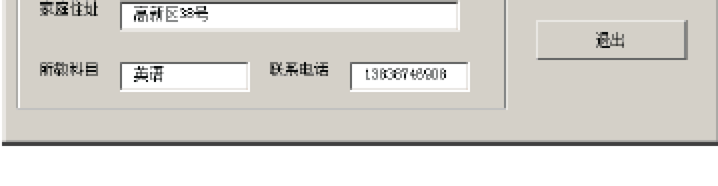



图 21.20　提示未输入职工编号



图 21.21　提示输入了相同姓名

（5）正确设置职工信息后，将信息写入工作表，程序提示写入成功，如图 21.22 所示。新的项目添加到工作表的最后一行，如图 21.23 所示。



图 21.22　输入成功的提示

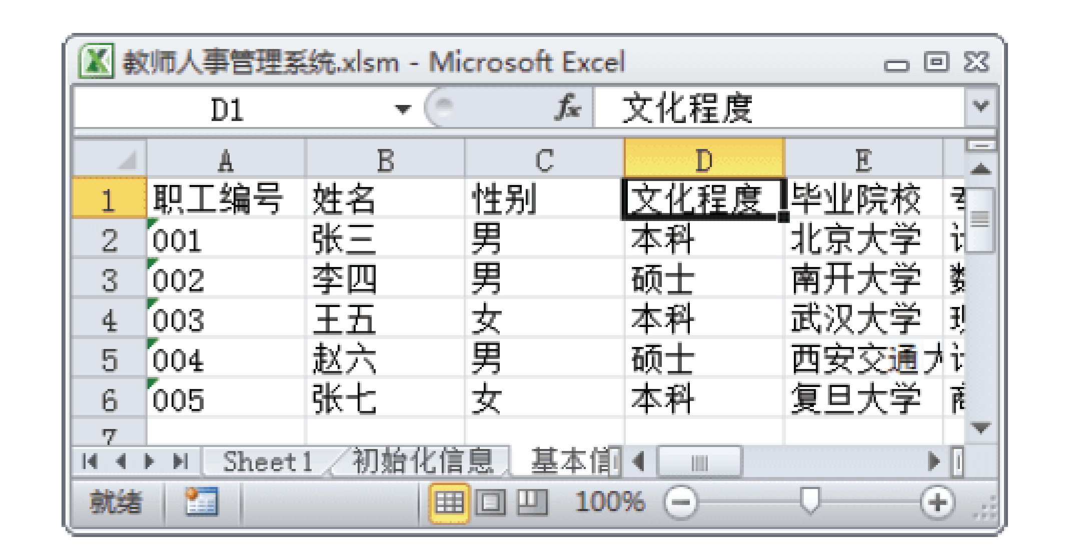


图 21.23　新项目添加到工作表最后一行

21.3.3　实现查询和修改数据功能

这里按照姓名来查询职工人事信息。在窗体的“姓名”文本框中输入需要查询职工姓名，在“基本信息表”中查询到匹配的姓名后，窗体控件显示相应的内容，实现功能的程序流程如图 21.24 所示。

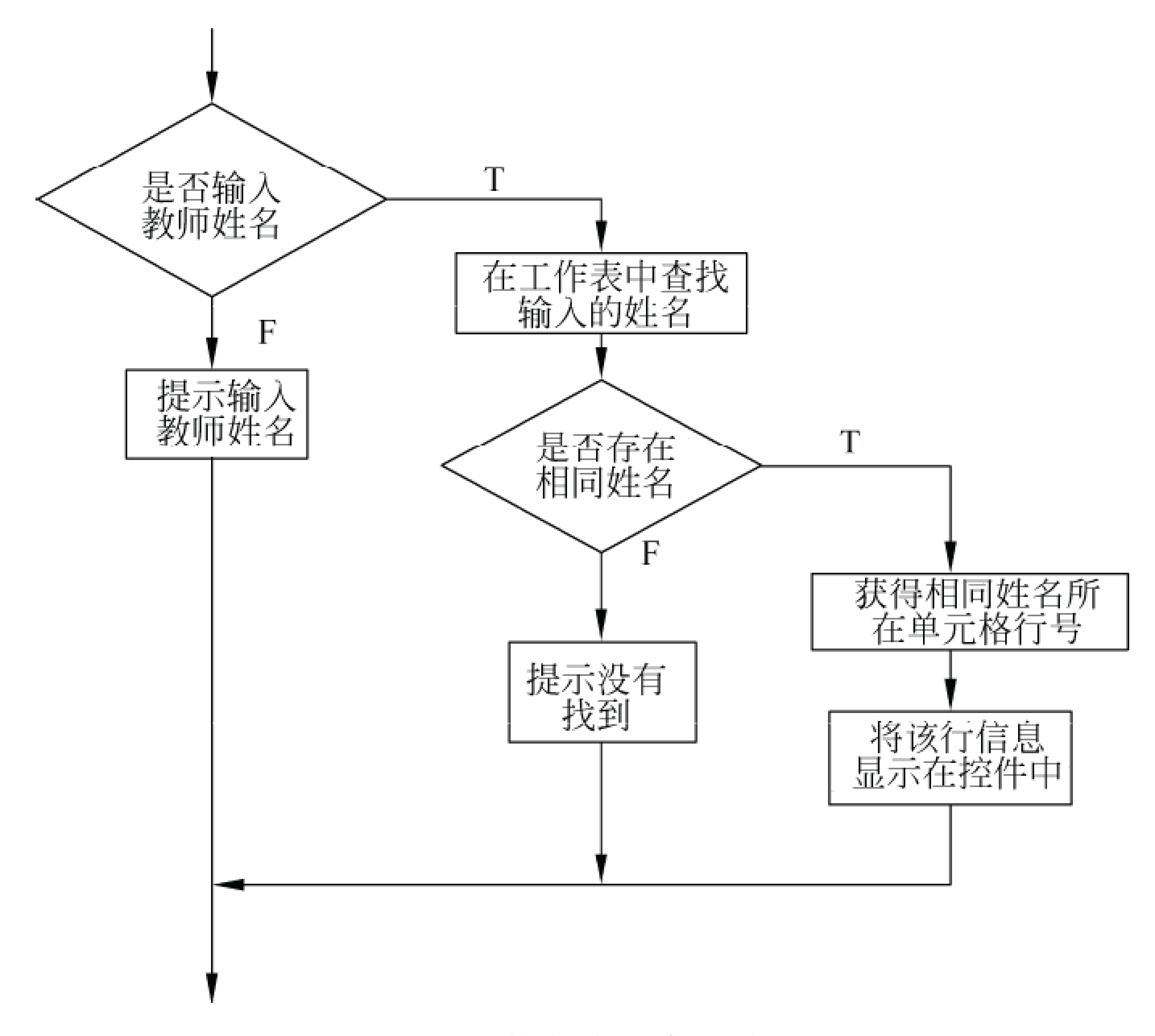


图 21.24　查询功能实现流程图

对于查询到的工作表中的信息，可以对项目数据进行修改，修改完成后将修改的结果重新写入工作表以实现对数据的更新。实现数据修改功能的程序流程如图 21.25 所示。下面介绍具体的制作步骤。

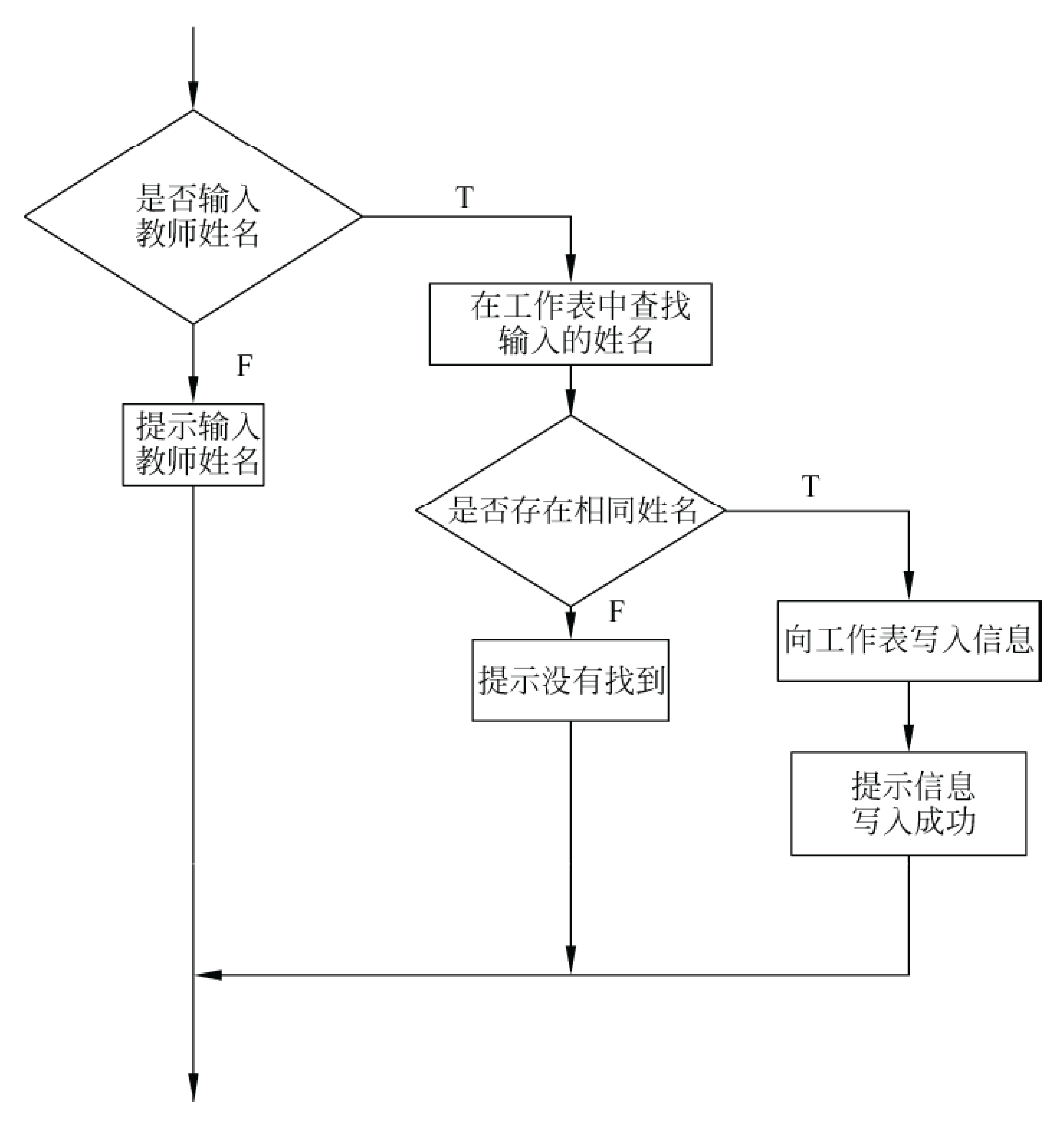


图 21.25　数据更新功能实现的流程图

（1）双击用户窗体中的“查询”按钮为按钮添加 Click 事件代码。按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton3_Click()
    Dim n As Integer
    Dim rng As Range
    If UserForm1.TextBox2.Text = "" Then                '判断是否输入姓名
        MsgBox "未输入职工姓名，请输入！"                 '给出提示
        Userform2.TextBox2.SetFocus                     '文本框获得焦点
        Exit Sub                                        '退出过程
    End If
    Set rng = Sheets("基本信息表").Columns(2).Find(TextBox2.Text)
                                                        '按姓名查找
    If Not rng Is Nothing Then                          '若找到需要的姓名
        n = rng.Row                                     '获得行号
        With Sheets("基本信息表")
            UserForm1.TextBox1.Text = .Cells(n, 1)          '写入编号
            UserForm1.TextBox3.Text = .Cells(n, 10)         '写入工作年限
            UserForm1.TextBox4.Text = .Cells(n, 5)          '写入毕业学校
            UserForm1.TextBox5.Text = .Cells(n, 6)          '写入所学专业
            UserForm1.TextBox6.Text = .Cells(n, 9)          '写入家庭住址
            UserForm1.TextBox7.Text = .Cells(n, 11)         '写入所教科目
            UserForm1.TextBox8.Text = .Cells(n, 8)          '写入联系电话
            If .Cells(n, 3) = "男" Then UserForm1.OptionButton1 _
```

```
21                .Value = 1                                          '性别设置为“男”
22                If .Cells(n, 3) = "女" Then UserForm1.OptionButton2 _
23                .Value = 1                                          '性别设置为“女”
24                UserForm1.ComboBox1.Value = .Cells(n, 4)    '设置文化程度
25                UserForm1.ComboBox2.Value = .Cells(n, 12)   '设置担任职务
26                UserForm1.ComboBox3.Value = .Cells(n, 7)    '设置职称
27            End With
28        Else
29            MsgBox "没有查询到需要的人员！"                          '提示没有找到
30        End If
31    End Sub
```

提示： 在这段代码中，第 04～07 行代码判断“姓名”单元格是否为空，如果为空，则给出提示。第 09 行代码在工作表的“姓名”列中查找相匹配的姓名，如果找到需要的姓名，第 11～27 行的代码将匹配姓名所在行的各个单元格中内容在窗体的控件中显示出来。如果没有找到匹配内容，则给出提示。

（2）当单击“查询”按钮时，程序首先判断“姓名”文本框是否为空，如果不为空，则将在“基本信息”工作表中查询“姓名”文本框中的姓名。如果找到匹配的姓名，将在窗体的控件中显示该项目对应的详细信息，如图 21.26 所示。如果没有找到对应的信息，则将提示人员没有找到，如图 21.27 所示。



图 21.26　显示查询结果

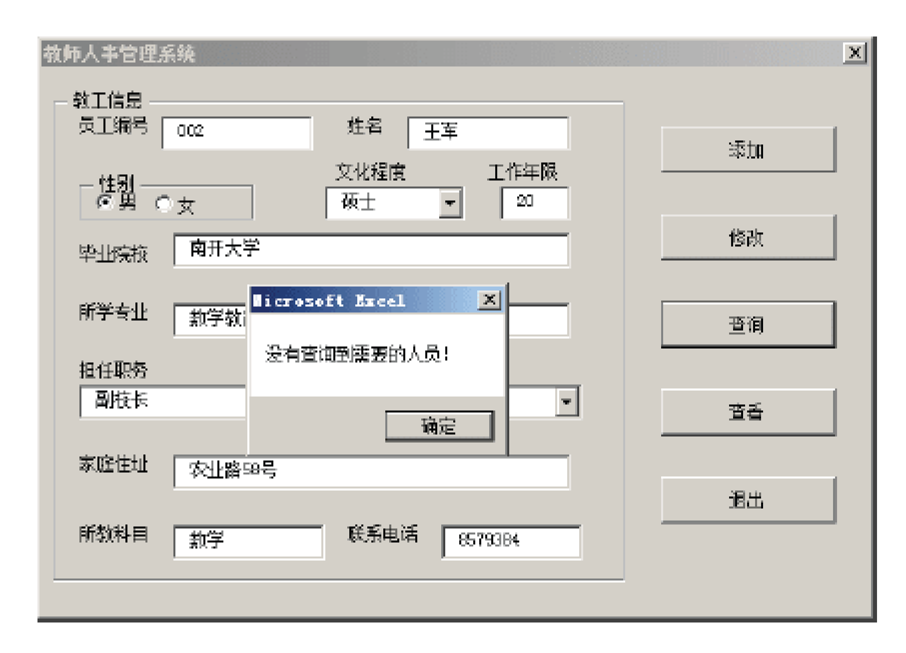


图 21.27　没有查找到需要人员时的提示

（3）在查询到需要的人员信息后，可以对获得的人员信息的任意一项进行修改，修改完成后单击窗体中的“修改”按钮，将新的信息写入工作表替换原有的资料，如图 21.28 所示。

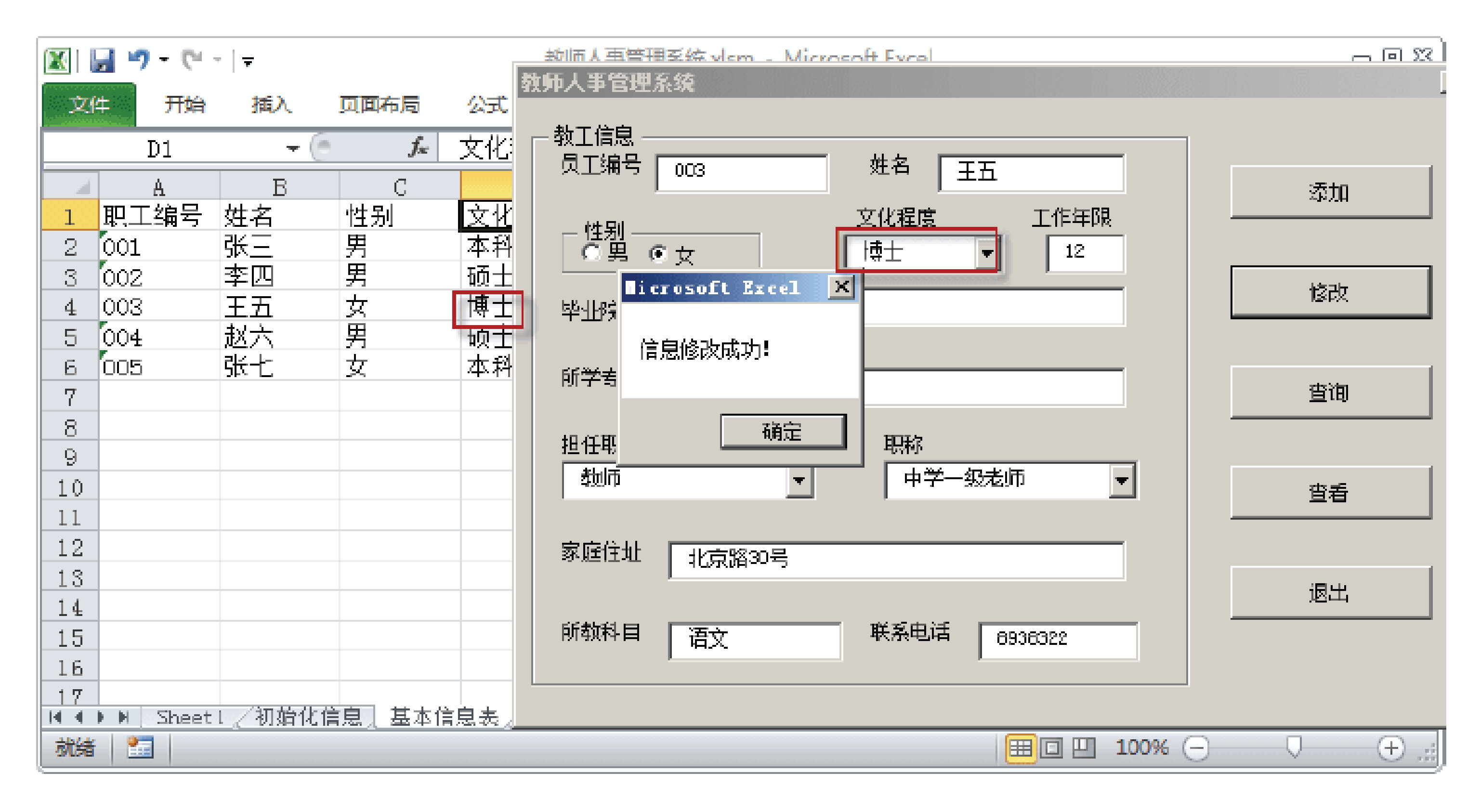


图 21.28　修改人员信息

这里，可以通过为“修改”按钮添加 Click 事件代码来实现人员信息修改，具体的程序代码如下所示：

```
01  Private Sub CommandButton2_Click()
02      Dim n As Integer
03      If UserForm1.TextBox2.Text = "" Then          '判断是否输入姓名
04          MsgBox "请输入需要修改资料的职工姓名！"     '提示输入姓名
05          TextBox1.SetFocus                         '文本框获得焦点
06          Exit Sub                                  '退出过程
07      End If
08      n = Sheets("基本信息表").Columns(2).Find(TextBox2.Text).Row
                                                      '工作表中查找相同姓名
09      If n <> 1 Then                                '如果找到相同姓名
10          UpdateD Sheets("基本信息表"), n            '写入信息
11          MsgBox "信息修改成功！"                    '提示修改成功
12      Else
13          MsgBox "信息表中没有此人记录！"             '提示没有找到指定人员
14          TextBox2.SetFocus                         '文本框获得焦点
15          Exit Sub                                  '退出过程
16      End If
17  End Sub
```

提示：在这段代码中，第 08 行按姓名在工作表中进行查询，并将找到的相匹配姓名的行号赋予变量 n。如果 n 不为 1，说明工作表中存在匹配项目，此时调用 Update 过程向工作表中写入新的内容实现数据更新。如果没有找到匹配的姓名，则给出提示。

21.3.4　实现退出程序和查看工作表

为了保护职工信息资料的安全，可以使工作簿的“初始化信息表”工作表和“基本信息表”工作表在操作时不可见。当需要对工作表进行维护而查看这些工作表时，可以通过单击“查看”按钮来显示它们。为了避免无关人员查看工作表，查看工作表需要密码验证功能。在完成操作后，单击“退出”按钮将退出工作簿。下面介绍具体的制作步骤。

（1）添加工作簿 Open 事件代码。工作簿 Open 事件代码如下所示：

```
Private Sub Workbook_Open()
    Sheets("基本信息表").Visible = False          '基本信息表不可见
    Sheets("初始化信息").Visible = False          '初始化信息不可见
    Sheets("sheet1").Activate                     '激活“sheet1”工作表
    UserForm1.Show                                '显示用户窗体
End Sub
```

警告：这里要注意一个问题，在隐藏工作表时，工作簿中必须要有一个可见的工作表。否则，在设置工作表的 Visible 属性为 False 时，VBA 会提示“不能设置类 worksheet 的 Visible 属性”错误，此时程序将无法执行。另外，如果对工作表进行了保护，在未解除保护前，设置属性也将出现相同的提示。

该事件代码使工作簿加载时，只显示一个空的工作表，而原有的“初始化信息”和“基本信息表”工作表将不可见，屏幕上将只显示用户窗体，如图 21.29 所示。

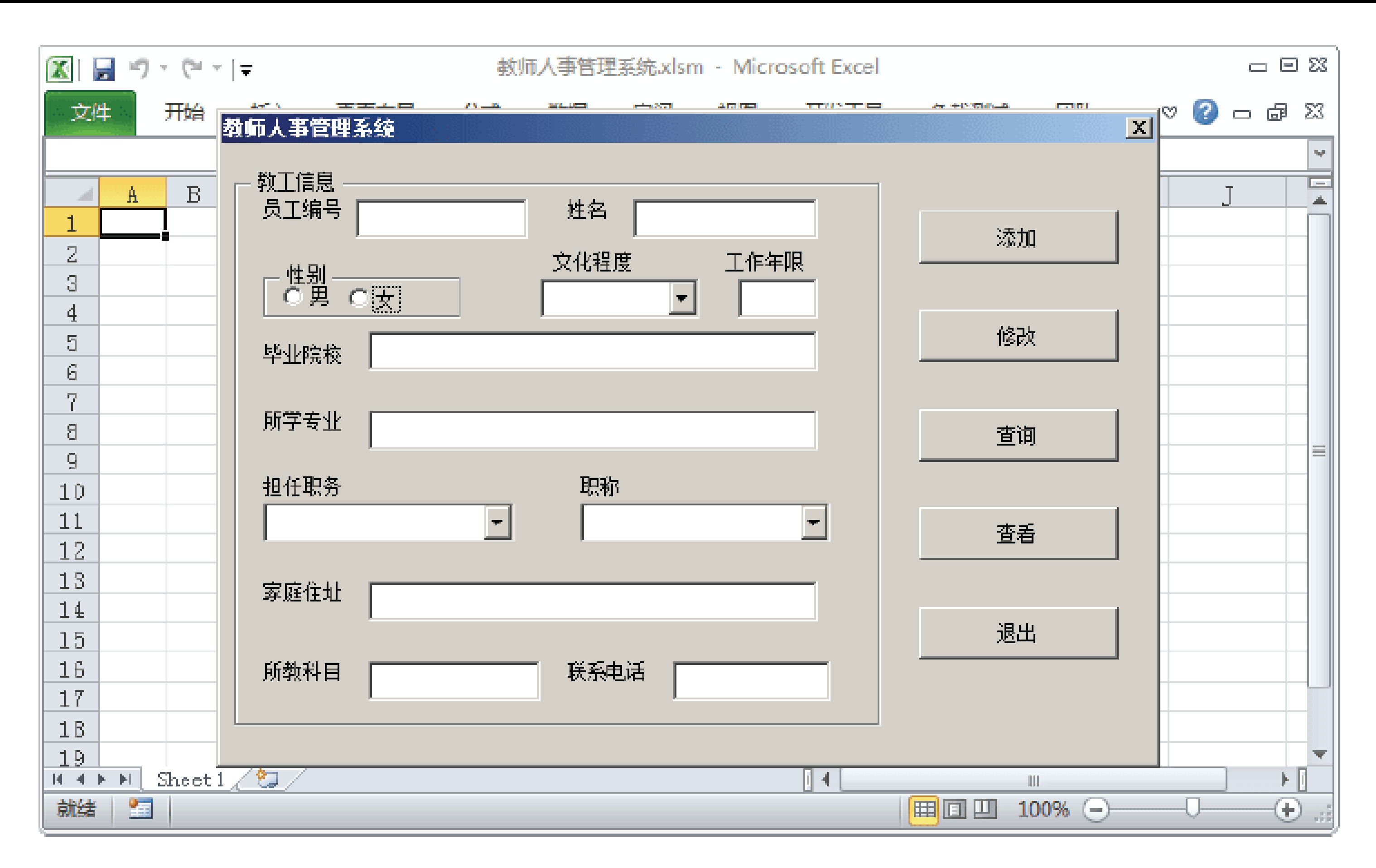


图 21.29　只显示用户窗体

（2）为用户窗体中的“查看”按钮添加 Click 事件代码。“查看”按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton4 Click()
    p = InputBox("请输入权限密码")                              '要求输入密码
    If p = "123456" Then                                        '如果密码正确
        Unload Me                                               '卸载窗体
        Sheets("基本信息表").Visible = True                     '显示“基本信息表”
        Sheets("初始化信息").Visible = True                     '显示“初始化信息”工作表
        Sheets("基本信息表").Activate                           '激活“基本信息表”工作表
    Else
        MsgBox "密码输入错误，您不能查看工作表。"               '提示没有查看权限
    End If
End Sub
```

提示： 这里，首先获取用户输入的密码值，将其与设定的密码“123456”进行比较，如果吻合，则卸载窗体并显示工作表。如果密码输入错误或未输入，则提示密码错误。

当单击该按钮时，首先要求输入权限密码，如图 21.30 所示。密码输入正确，窗体将关闭，隐藏的工作表将显示。密码输入错误或没有输入密码都将给出提示，关闭提示对话框后将返回用户窗体，如图 21.31 所示。

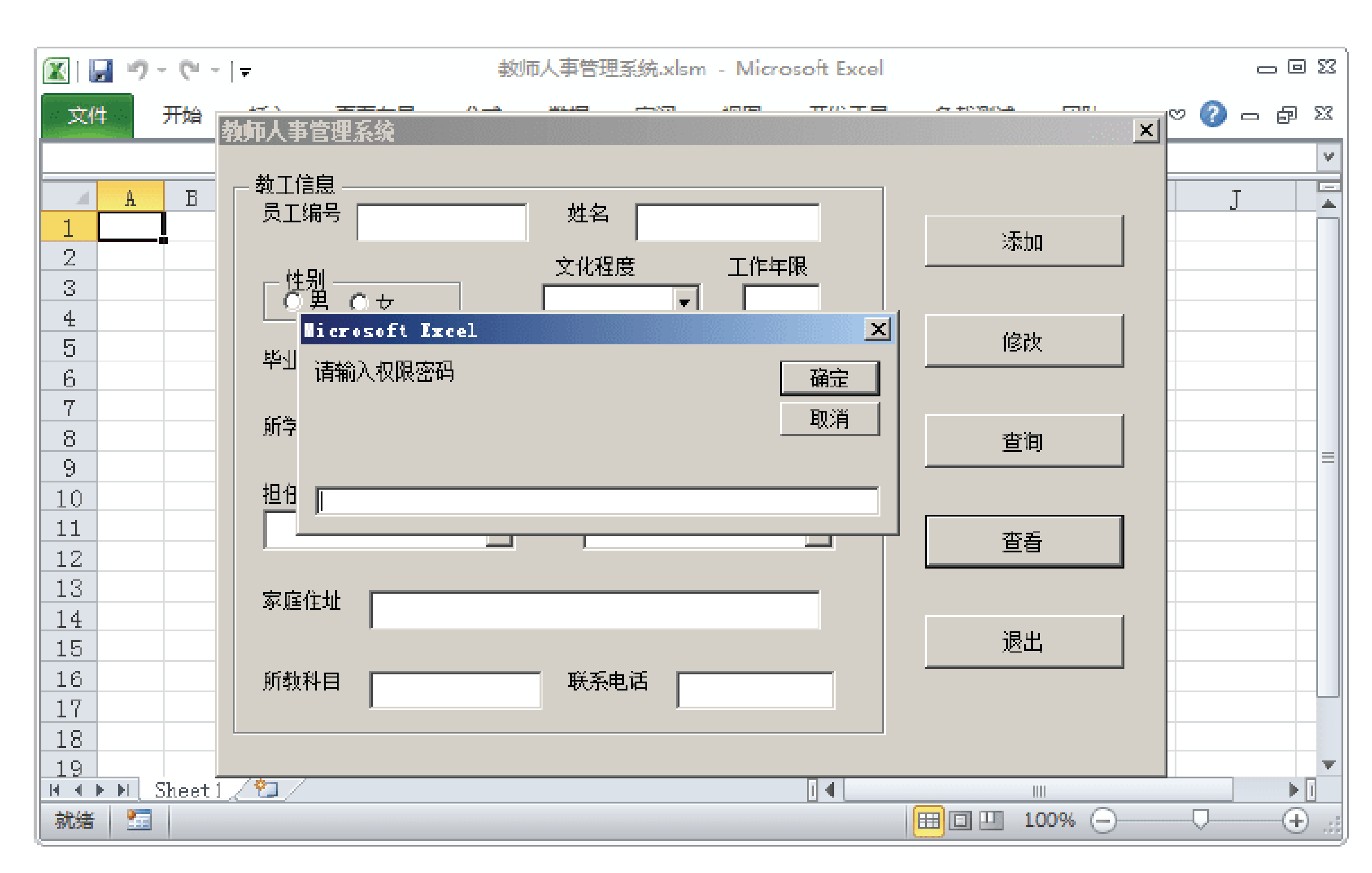


图 21.30　要求输入权限密码

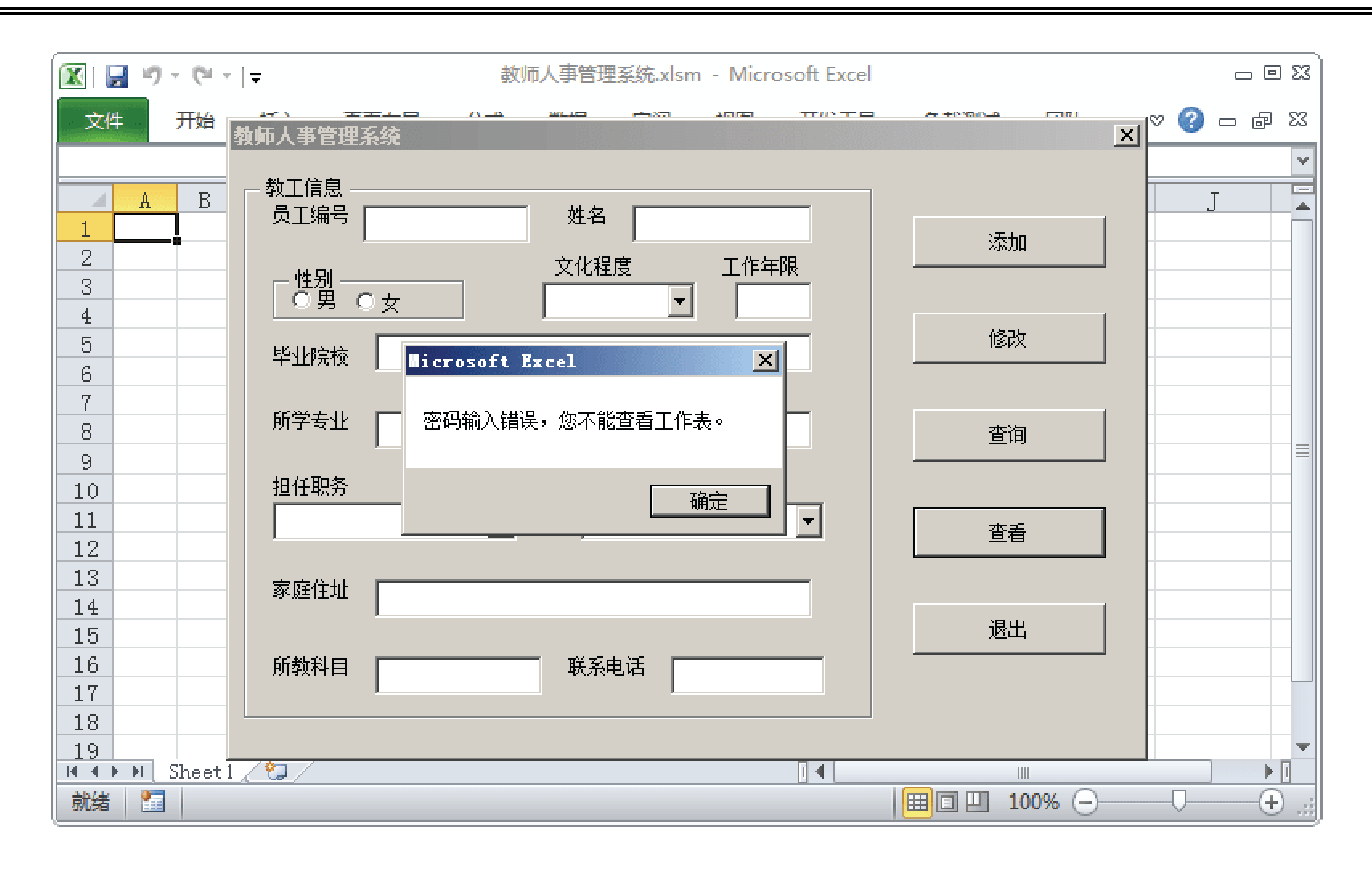


图 21.31　密码错误时的提示

（3）为“退出”按钮添加 Click 事件代码实现退出程序的功能。“退出”按钮的 Click 事件代码如下所示：

```
Private Sub CommandButton5_Click()
    a = MsgBox("真的要退出系统吗？", vbOKCancel)          '提示是否退出系统
    If a = vbCancel Then                                  '如果单击“取消”按钮
        Exit Sub                                          '退出过程
    Else
        Workbooks.Close                                   '关闭工作簿
    End If
End Sub
```

提示：这里，退出时应该允许用户对是否退出进行选择，使用 If 结构来判断用户的选择，如果单击的是提示对话框中的“取消”按钮，则关闭对话框而不退出系统。否则，关闭将工作簿。

代码实现两个功能，首先提示是否真的退出系统，如图 21.32 所示。如果单击提示对话框中的“确定”按钮，则关闭工作簿，否则返回用户窗体。

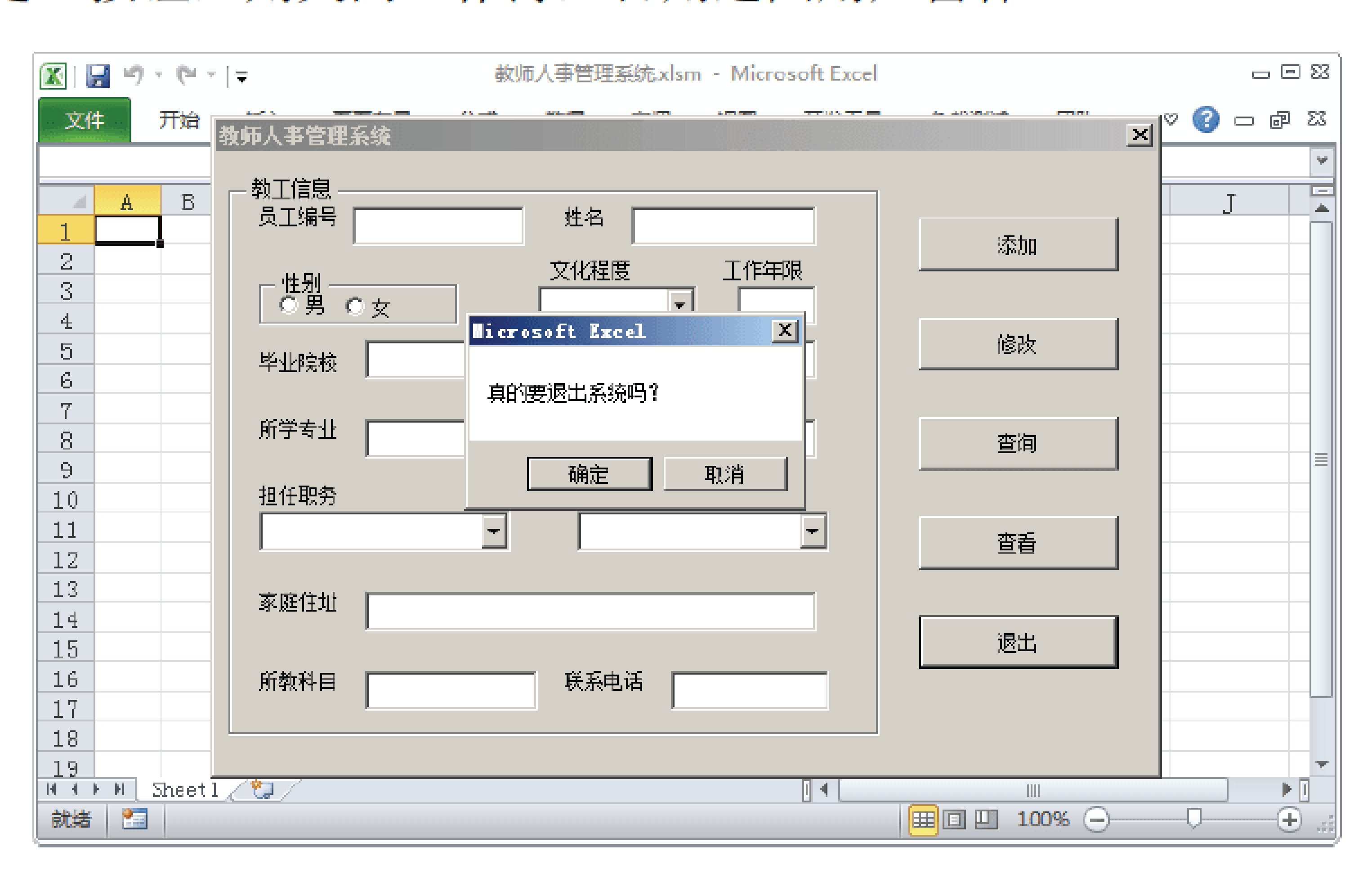


图 21.32　提示是否退出系统

21.4　小　　结

通过本实例的制作，读者将能进一步熟悉用户界面的设计方法、事件驱动的程序设计理念以及工作簿、工作表和单元格的操作方法。完成本实例的制作后，读者将能够掌握下面的编程技巧：

本实例通过触发相应的事件来启动事件代码实现程序的各项功能。通过本实例的制作，读者将能够掌握“命令按钮”控件的鼠标单击事件（Click 事件）的使用方法、窗体的初始化事件（Initialize 事件）在控件初始化时的意义以及工作簿的 Open 事件的使用技巧，读者也将能够进一步熟悉控件的事件程序设计思路和编写技巧。

本实例需要将控件的输入内容写入工作表的单元格中，在查询时也需要将指定单元格在控件中显示出来。通过本实例的制作，将能够进一步熟悉工作表和单元格的引用方法，掌握窗体中控件的引用和属性的设置方法。

使用 Find 方法能够在工作表中查找指定的内容，这是本例查询功能和检验输入是否重复的关键。通过本实例的制作，读者将了解如何使用 Find 方法获得匹配内容所在的行、列和单元格，并掌握对查找到的数据进行处理的方法。通过设置工作表对象的 Visible 属性可以实现工作表的隐藏和显示，这是使用自定义用户界面时保护工作表的一种有效的方法。